USING BIOMECHANICAL CONSTRAINTS TO IMPROVE VIDEO-BASED MOTION CAPTURE

THÈSE N^{O} 2892 (2003)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut des systèmes informatiques et multimédias

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Lorna HERDA

informaticienne diplômée de l'Université de Genève
et de nationalité allemande

acceptée sur proposition du jury:

Prof. A. Fua, directeur de thèse
Dr R. Horaud, rapporteur
Prof. D. Thalmann, rapporteur
Prof. L. Van Gool, rapporteur

Lausanne, EPFL
2003

Abstract

In motion capture applications whose aim is to recover human body postures from various input, the high dimensionality of the problem makes it desirable to reduce the size of the search-space by eliminating *a priori* impossible configurations. This can be carried out by constraining the posture recovery process in various ways. Most recent work in this area has focused on applying camera viewpoint-related constraints to eliminate erroneous solutions. When camera calibration parameters are available, they provide an extremely efficient tool for disambiguating not only posture estimation, but also 3D reconstruction and data segmentation.

Increased robustness is indeed to be gained from enforcing such constraints, which we prove in the context of an optical motion capture framework. Our contribution in this respect resides in having applied such constraints consistently to each main step involved in a motion capture process, namely marker reconstruction and segmentation, followed by posture recovery. These steps are made inter-dependent, where each one constrains the other.

A more application-independent approach is to encode constraints directly within the human body model, such as limits on the rotational joints. This being an almost unexplored research subject, our efforts were mainly directed at determining a new method for measuring, representing and applying such joint limits. To the present day, the few existing range of motion boundary representations present severe drawbacks that call for an alternative formulation. The joint limits paradigm we propose not only overcomes these drawbacks, but also allows to capture intra- and inter-joint rotation dependencies, these being essential to realistic joint motion representation. The range of motion boundary is defined by an implicit surface, its analytical expression enabling us to readily establish whether a given joint rotation is valid or not. Furthermore, its continuous and differentiable nature provides us with a means of elegantly incorporating such a constraint within an optimisation process for posture recovery. Applying constrained optimisation to our body model and stereo data extracted from video sequence, we demonstrate the clearly resulting decrease in posture estimation errors. As a bonus, we have integrated our joint limits representation in character animation packages to show how motion can be naturally constrained in this manner.

Version abrégée

Dans le contexte des applications de capture de mouvement dont le but est d'inférer la posture de modèles de corps humain, la haute dimensionalité du problème implique qu'il est souhaitable de réduire la taille de l'espace de recherche en éliminant des solutions *a priori* impossibles. Ceci peut être réalisé en contraignant le processus de détermination de la posture de diverses manières. La majorité des récents travaux dans ce domaine a focalisé sur l'imposition de contraintes liées aux points de vue des caméras dans le but d'éliminer les solutions erronées. Quand les paramètres de calibration des caméras sont disponibles, ils présentent un outil extrêmement efficace pour éliminer les ambiguités, et ceci pas seulement dans le cadre de l'estimation de la posture, mais aussi lors de la reconstruction 3D et de la segmentation des données.

Un gain en robustesse certain s'obtient en effet par l'application de telles contraintes, ce que nous démontrons dans le cadre de la capture de mouvement optique. Notre contribution dans ce domaine réside dans le fait d'avoir appliqué de telles contraintes de le manière consistante à chaque étape qu'implique le processus de capture de mouvement, c'est-à-dire la reconstruction des marqueurs et leur segmentation, suivies de l'estimation de la posture. Ces étapes sont rendues inter-dépendantes, chacune contraignant l'autre.

Une approche plus indépendente vis-à-vis de l'application consiste à encoder des constraintes directement dans le modèle de corps humain utilisé, sous forme, par exemple, de limites sur les rotations des articulations. Ceci étant un axe de recherche quasiment inexploré, nos efforts ont principalement porté sur la mise au point d'une nouvelle méthode pour mesurer, représenter et appliquer de telles limites sur l'espace de mouvement d'une ou plusieurs articulations. A ce jour, les quelques rares représentations de limites articulaires existantes présentent de sérieux désavantages, appelant ainsi le besoin d'une formulation alternative. Le paradigme que nous proposons ne se contente pas de pallier ces inconvénients, mais permet aussi de capturer les dépendences intra- et inter-articulaires, celles-ci étant primordiales à une représentation réaliste des mouvements articulaires. L'espace de mouvement d'une articulation est défini par une surface implicite, son expression analytique nous permettant aisément de déterminer si une rotation est valable ou non. De plus, son caractère continu et différentiable nous fournit un moyen d'incorporer élégamment une telle contrainte dans un processus d'optimisation en vue d'estimer la posture. En appliquant une optimisation contrainte à notre modèle de corps humain et à des données stéréo extraites de séquences vidéo, nous démontrons la nette décroissance des erreurs d'estimation. Nous avons également integré notre représentation de limites articulaires dans des applications d'animation afin de montrer à quel point le mouvement peut être naturellement contraint de cette manière.

Acknowledgements

> "Every particle in the universe affects every other particle, however faintly or obliquely. Everything interconnects with everything. The beating of butterfly wings in China can affect the course of an Atlantic hurricane."
>
> -Douglas Adams, *The Long Dark Tea-Time Of The Soul*

Chaos Theory applied to the universe by transitivity also holds true in the case of a PhD thesis, so I'd like to start by implicitly thanking all the people who contributed in one way or another, whether it seemed infinitesimal at the time or not.

On a more tangible level, I am grateful to Prof. Thalmann for having accepted me within his lab, where I remained up to the day the Computer Vision Lab (peacefully) gained its independence. I would also like to extend my gratitude to the members of the defence committee, Profs. Horaud, Van Gool and Thalmann, for the care they took in reviewing the present work.

I would like to thank the two persons who forever changed the face of this PhD, starting with Prof. Hanson, whose overwhelming enthusiasm for quaternions brought a whole new dimension (!) and direction to this endeavour. The second person would be my colleague and friend, Raquel Urtasun. I owe a lot to her always inquisitive mind and steady flow of useful comments and suggestions. I am also in debt to her as my endless source of motivation - there was always enough to go around, even when the going got tough.

For enlightening input and help in their respective areas of expertise, I am grateful to Ronan Boulic, Benoît LeCallennec and my former colleagues Ralf Plänkers, Amaury Aubel and Paolo Baerlocher.

I would also like to extend my thanks to:

- Josiane Gisclon, and the system administrators, Etienne de Sevin, Jan Ciger and Slobodan Ilic, for their general efficiency and helpfulness;
- Emilio Casanova for his help with the camera set-up and visualisation software;
- Tom Molet at Miralab, for years of technical assistance with the Vicon motion capture system;
- Raquel, Emilio, Nicolas Magnin and Pascal Glardon, the motion capture subjects, who gracefully accepted the idea of being clad in a diving suit and decorated like Christmas trees, all in the name of scientific research;
- Awen Limbourg and Olivier Yglesias, the motion capture photo models;
- Mireille Clavien and Ali Shahrokni for creating the keyframe sequences;
- my past and present colleagues from the Computer Vision and Virtual Reality Labs, for all the good times.

On a less professional level, I would like to thank my friends and family for having been there always, especially Nicolas, my "significant other", for his active fight against those inevitable moments of doubt. Finally, I would like to thank my mum, the greatest person in the world, and always a full-time supporter of everything I did.

I would like to wrap up this acknowledgements section by sincerely thanking my PhD adviser, Prof. Fua, for his unfailing guidance and support through all these years. As a parabola says more than a thousand words (yes, I made that one up), I conclude with a well-known tale, which also sums up a wise remark my friend Dominik made the day I sold my soul to the devil and signed up for a PhD.

The Thesis One sunny day, a rabbit came out of her hole in the ground to enjoy the fine weather. The day was so nice that she became careless and a fox sneaked up behind her and caught her.

"I am going to eat you for lunch!" said the fox.
"Wait!" replied the rabbit, "You should at least wait a few days."
"Oh yeah? Why should I wait?"
"Well I am just finishing my thesis on 'The Superiority of Rabbits over Foxes and Wolves'."
"Are you crazy? I should eat you right now! Everybody knows that a fox will always win over a rabbit."
"Not really, not according to my research. If you like, you can come into my hole and read it for yourself. If you are not convinced, you can go ahead and have me for lunch."
"You really are crazy!"

But since the fox was curious and had nothing to lose, he went with the rabbit. The fox never came out. A few days later the rabbit was again taking a break from writing and sure enough, a wolf came out of the bushes and was ready to set upon her.

"Wait!" yelled the rabbit, "You can't eat me right now."
"And why might that be, my furry appetiser?"
"I am almost finished writing my thesis on 'The Superiority of Rabbits over Foxes and Wolves'."
The wolf laughed so hard that he almost lost his grip on the rabbit.
"Maybe I shouldn't eat you, you really are sick in the head. You might have something contagious."
"Come and read it for yourself, you can eat me afterwards if you disagree with my conclusions."

So the wolf went down into the rabbit's hole and never came out. The rabbit finished her thesis and was celebrating in the local lettuce patch. Another rabbit came along and asked,

"What's up? You seem very happy."
"Yup, I just finished my thesis."
"Congratulations. What's it about?"
"'The Superiority of Rabbits over Foxes and Wolves'."
"Are you sure? That doesn't sound right."
"Oh yes. Come and read it for yourself."

So together they went into the rabbit's hole. As they entered, the friend saw the typical graduate abode, albeit a rather messy one after writing a thesis. The computer with the controversial work was in one corner. And to the right there was a pile of fox bones, on to the left, a pile of wolf bones. And in the middle was a large, well-fed lion. The moral of the story:
... the title of your thesis doesn't matter,
... the subject doesn't matter,
... the research doesn't matter,
... all that matters is who your adviser is.

Contents

Chapter 1

Introduction

The constant technological leaps of the past 30 years in the computer graphics and vision domain have led our everyday world to be populated with a plethora of animated virtual humans and other creatures, with increasingly flawless skin textures and fluid motion. What was not long ago considered to be a scientific achievement is now taken for granted, at least by the public. But is the book of realistic human animation really closed?

We will argue that it is not, and we will focus on what is the currently most popular tool for recovering human motion to make our point. The tool in question is motion capture, and it is nowadays the most wide-spread means for recovering motion from live subjects and transferring it to virtual models. In the current work, we will focus on motion capture techniques that recover human motion from image sequences using synchronised cameras, thus yielding dynamic 3D data. When recovering postures from such data, a common issue is that the state-space of possible solutions is vast and therefore needs to be pruned in order to eliminate ambiguous and physically impossible postures.

One way to do this is to impose constraints such as consistency with camera viewpoint, or temporal smoothing. We will demonstrate that this results in an increased robustness by applying such constraints to an optical motion capture framework. The constraints have been integrated into each of the various motion capture steps, these being marker reconstruction and tracking, followed by estimation of the motion parameters. The novelty of the approach resides in the inter-dependency we have created between these steps, so as to impose a constraint of mutual coherence, instead of dealing with each sub-task separately, as is generally the norm.

Another way to prune the state-space is to limit the motion of the virtual model itself, for example by applying joint limits, collision detection and dynamic constraints. We have decided to focus on the most challenging of the three, namely joint limit constraints. Collision avoidance is a trivial task in this case, and human dynamics have already been vastly explored by the scientific community, but joint limit representations remain to this date simplistic and insufficient. In this context, we have inferred, represented and integrated such joint limit constraints into a motion capture process using stereo data extracted from video sequences, thereby improving posture recovery by eliminating physically impossible solutions. In a nutshell, we propose a means of representing a range of motion, not only for single joints, but also for coupled joints, in the aim of improving motion recovery and character animation.

In practice, we have handled the two constraint categories separately, in order to assess their respective contributions to motion capture performance, each within a framework where their application is particularly relevant.

In this introductory chapter, we present what we have said to nowadays be the most popular means for animating virtual characters, namely motion capture. The need to recover human motion arises at many different levels, the various possible applications being briefly reviewed. From there on, we lay out the various obstacles that need to be overcome in this particular context, and finally, we describe what our contribution to the area is and what improvements are brought on by the present work.

Figure 1.1: Virtual human character in the movie feature "Final Fantasy" by SquareSoft.

1.1 Underneath it all

At the present day, it is an indisputable that animated features such as SquareSoft's "Final Fantasy" are productions resulting from the bringing together of all available top state-of-the-art technology. Only the greatest cynic would say that it lacks in visual realism and that "everything is still to be done" (Figure 1.1).

It is not so, but nevertheless, for those who stay on after the movie to read the credits display, it should be duly noted that the number of animators needed for producing such a movie feature forms a respectable armada. The animators need to be given the credit of having driven the illusion to the state of near perfection that it has now reached, where every ripple on the skin is painstakingly refined until it no longer shocks our eye or disappoints our expectations. However, if motion is nowadays so easily captured from a live human source and transposed onto a virtual human being, wouldn't it be enough for an animator to design the virtual character and then to animate it automatically using this previously recorded data? When watching the "Making of" features of the such movies, it actually does sound that easy. Therefore, it should be pointed out that:

1. recorded motion data needs a lot of processing, before, during and after, in order to make it usable for virtual character animation;

2. resulting animations need to be frequently re-touched by animators, as the human eye is unforgiving when it comes to detecting unrealistic human representations.

Recording motion data from a live source is referred to as "motion capture" and given the mind-boggling amount of scientific literature available on the subject, it is definitely a hot research topic. The quality of motion re-targeting, which means applying the captured motion to a virtual character, is directly linked to the quality of the motion capture itself. Indeed, details that have not been captured cannot be ultimately re-created. The human body being an extremely complicated structure, this puts an extra weight on the frail shoulders of the human motion capture process.

It would be a legitimate question to ask whether motion capture is worth the while from a commercial point of view if so much work is involved. Motion capture is to character animation what computers are to bureaucracy: the number of people that need to be employed does not decrease, but their efficiency should (theoretically) increase. Indeed, it is difficult to imagine how a generated motion could be more realistic than if it were directly transferred from a recorded live motion! However, as long as the motion capture processing overhead has not been sufficiently reduced to make it appealing in terms of money and time, many animations will still be created using the regular key-framing technique. This involves an animator defining some key postures

and the intermediate postures being automatically computed by interpolation. This technique does of course not guarantee realism, which can only be enforced by the keen eye of the animator and a lot of fine-tuning, as not only do the key postures need to look realistic, but also the generated intermediate frames.

To spare the animators this harassing task, various motion capture techniques exist and will be presented in an upcoming section, along their respective advantages and short-comings. In the sections that follow, we start with the practical application of motion capture. Of all the existing motion capture techniques, the camera-based ones are those presenting the greatest challenge and are currently the centre-point of most on-going research. We will therefore also be focusing exclusively on camera-based motion capture, and move on to discussing its inherent difficulties and finally, what we intend our contribution to be in this respect.

1.2 Applications of motion capture

The reader may now be under the impression that Hollywood is the reason to be of motion capture. It is true that the entertainment sector has cut itself the largest part of the pie, but other applications exist.

The first is clinical studies, where motion captured biomechanical data is used for gait analysis or orthopaedic applications, such as joint mechanics, analysis of the spine, prosthetic design and sports medicine. Gait analysis is useful to measure a degree of change in conditions such as arthritis or strokes. Such data analysis can also be used to determine treatment for pathological conditions that affect the way we walk, such as cerebral palsy, as well as for rehabilitation purposes, for patients with knee, pelvis or ankle problems [Menache1999].

An increasingly popular application domain is sports, the aim being to improve the performance of athletes by analysing the captured motion and comparing it with that of recognised professionals, thus detecting and correcting eventual weaknesses. Tennis, gymnastics and swimming athletes are typical customers for such motion capture systems, golf being one of the most popular sport to benefit from such high-tech aids, such as the software packages sold by Sports Motion Inc. or Dartfish's DartGolfer, but video coaching is proposed for a good two dozens of other sports.

Crash simulations are another field of application where animated characters play the role of crash test dummies, and are the only possibility for effective safety testing (or so we hope).

Finally, we return to the entertainment field, where motion capture has its widest range of application. Computer graphics animation features are of course the most obvious use of animated characters, but traditional movies also increasingly employ virtual actors, either for animating fantastic creatures, as in "Lord of the Rings", or for adding digital extras, such as crowds in "Titanic".

Video game companies also represent a vast share of the motion capture marker, where"typical" moves can even be bought key-in-hand from specialised motion capture studios such as House of Moves or Imagination In Motion. Such a solution is more cost-effective for gaming companies than traditional character animation.

A still minor but also growing fields of application are teleconferencing and interactive virtual worlds, where in both cases a virtual avatar represents the real person, and is able to reproduce the gestures and expressions of its living counterpart.

Depending on the degree of precision that is necessary, and on the complexity of the captured moves, a different motion capture technique can be preferred in each case, the possible range of systems being presented in the next chapter.

1.3 Inherent difficulties of camera-based motion capture

In this section, we will assess the obstacles arising at each level of the motion capture processing flow, which can be sub-divided into the following items:

1. data acquisition and feature extraction;

2. data segmentation, that is, establishing the correspondence between image features and human body features;

3. posture estimation.

1.3.1 Data acquisition

At this level, the difficulty is linked to the nature of the captured data and the amount of information available about it. We can distinguish the following categories:

- marker data in calibrated infrared camera views,
- video sequences from calibrated or uncalibrated CCD cameras,
- synchronised video sequences from several calibrated CCD cameras,
- volumetric data.

The quality of the motion capture output is directly linked to the quality of the input image, which can vary depending on resolution, lighting conditions, contrast, etc.

In the case of marker data from infrared cameras, any object in the 3D scene not seen by at least two cameras will simply not be present in the reconstructed scene. Marker to skin motion is bound to introduce further measurement errors [Cappozzo *et al.*1996]. As to reconstruction from video sequences, it is inherently ambiguous due to the fact that a certain number of postures will project identically onto an image. Furthermore, the model will typically experience singularities when motion is carried out in the direction perpendicular to the view point axis [Rehg *et al.*2003]. In the light of what precedes, it is surprising, as [Gavrila1999] underlined in his survey, that not more research work is directed towards motion capture using 3D data, despite the bulkier aspect of the hardware setup. Some research work has focused on fitting a body model to volumetric data, such as (Bottino et al., 1998) and (Mikić et al., 2003), but this requires an even heavier and more sophisticated setup. In terms of simple cameras, apart, obviously, from the couple of papers on optical motion capture, only [Plänkers and Fua2001] and [Demirdjian2003] seem to have taken interest in using their multi-camera setup to extract range data and directly fitting a body model to it. When using synchronised cameras, the singularities of the single camera case can be solved, but one still has to reckon with 3D reconstruction errors, for example when there is too little texture for reliable matching between views.

1.3.2 Data segmentation

To to derive a body posture from motion capture data, it is necessary to identify, in each frame, the extracted features and their correspondence with the significant items of the object whose pose we are trying to recover. In our case, we are recovering the posture of a human body representation, to which we will refer to as the human body model, and the significant items would either be the limb extremities, the various body parts, or the joints.

If the feature to identify is discreet and sparse, or continuous and rigid, frame-to-frame tracking is usually preferred, the identity of the feature thus being propagated from frame to frame. The disappearance of a part of the data due to occlusions often puts this process to the test, making error recovery schemes absolutely essential.

When using large amounts of discreet data such as 3D points from stereo data or continuous features such as silhouettes, the segmentation process will need to be carried out in each frame independently, thus avoiding eventual error propagation. In the case of silhouettes, the problem of occlusion once again rears its ugly head and seriously hampers the process. In the case of calibrated cameras, the problem is less drastic, as the intrinsic camera parameters provide valuable information for dealing with occlusion.

1.3.3 Pose recovery

After much toil, we may finally reach the last milestone of motion capture, namely actually recovering the correct model pose that corresponds to our extracted data features. A number of optimisation techniques are readily available for solving this problem, which more often than not will not have a unique solution. Indeed, it is almost inevitable that several postures will correctly correspond to the same data, due to the fact that the latter is usually ambiguous, and that the system may be over-parametrised.

From now on, we will also refer to a posture of the body model as a "state", the entire set of possible postures of a model being referred to as the "state-space vector". In practice, to infer the correct state of a model, the state-space will have to be pruned by eliminating the ambiguities, more precisely by throwing away all solutions that do not comply with a certain set of conditions, that we will call "constraints".

1.4 Contribution of this thesis

In this work, we explore the use of constraints to carry out reliable matching of a 3D model to dynamic 3D data. Posture recovery from 3D data remains a process of high-dimensionality. This implies that the state-space can be so large that, even given perfect data, it may be difficult to find the optimal solution without constraining the search. In practice, imposing constraints becomes even more critical because the data is often noisy and incomplete.

We propose novel ways to handle and represent constraints that restrict motion capture algorithms to humanly feasible configurations, thereby reducing the search space they must explore and increasing their reliability. We distinguish two kinds of constraints, extrinsic ones and intrinsic ones.

Intrinsic constraints are part of the model itself, thus defining a sub-space of valid postures with respect to the state-space defined by the joint geometry. Intrinsic constraints applied to motion capture to this day include:

- joint limits, which limit the range of motion of the joints;
- collision detection that ensures that body parts do not inter-penetrate;
- dynamics, such as gravity or balance.

Extrinsic constraints, by contrast, are provided by sources other than the body model. Typical ones are:

- motion smoothing that limits frame-to-frame joint velocity and acceleration;
- motion models, which apply learned motion patterns in an effort to eliminate postures that are inconsistent with respect to learned parameters;
- visibility constraints, based on the knowledge of the camera set-up configuration.

1.4.1 Visibility and motion constraints

In an optical motion capture application, the 3D data representing markers is sparse and valuable, mainly because the markers are usually located at joint positions, and therefore the correct 3D reconstruction and identification of each is vital to posture recovery.

The use of extrinsic constraints in this case aims not only at recovering the correct motion parameters of the model, but also to increase the robustness of the steps that precede posture recovery. Knowledge of the camera parameters, as well as their position and orientation, will enable us to estimate the measure of confidence of the 3D reconstruction, and correct it in the event that this measure is too low for comfort.

The 3D data then requires segmentation — or "identification" —, which is carried out by tracking the markers over the sequence, not only in 3D space but also in their original camera views, to take as much advantage as possible from camera calibration information. We will also dynamically establish the spatial relationship between

the data and the model, this acting as an additional constraint on the 3D reconstruction and segmentation, as an over-all coherence will need to be respected.

Temporal constraints are enforced on the marker tracking and posture estimation levels, to ensure that frame-to-frame speed and acceleration remain within acceptable ranges, the scheme implementing this constraint being based on a fluid dynamics algorithm.

In essence, neither camera-related nor temporal constraints are anything new in motion capture. However, even if it is true that such constraints have often been applied to each step of the process, whether reconstruction, segmentation or posture recovery, they have never been enforced in a manner that interleaves the various steps and constraints in a closed circuit. Our contribution consists in merging all modules into an inter-dependent process, where the current state of each component acts as a coherence constraint on the others. The human body model, in this case, not only serves the purpose of predicting or recovering a posture, but basically drives the entire process, hand in hand with the extrinsic constraints. We will show that our integrated algorithm allows to reduce over-all errors, during 3D reconstruction, segmentation, and subsequently, posture recovery.

1.4.2 Joint limits and their dependencies

Limiting the range of motion that can be recovered is currently done in many existing systems but the limits are usually represented in an over-simplified manner that does not closely correspond to reality. The most popular approach is to express them in terms of hard limits on the individual Euler angles used to parameterise joint rotations. This accounts neither for the dependencies between angular and axial rotations in ball-and-socket joints, such as the shoulder joint, nor those between separate joints such as the hip and knee. In other words, how much one can twist one's arm depends on its position with respect to the shoulder. Similarly, one cannot bend one's knee by the same amount for any configuration of the hip. As a result, the constraints being imposed are bound in such a way as to be inaccurate and, therefore, sub-optimal in terms of constraining the motion recovery. An additional difficulty stems from the fact that experimental data on these joint limits is surprisingly sparse: medical text books typically give acceptable ranges in a couple of planes—such as the so-called sagittal or coronal planes—but not for the whole configuration space which is what is really needed by an optimisation algorithm searching that space. Here, we first propose a quaternion-based model approach for explicitly representing and measuring dependencies between the three degrees of freedom of a ball-and-socket joint such as the shoulder. It relies on measuring the joint motion range using optical motion capture, converting the recorded values to joint poses encoded by a coherent quaternion field representation of the joint orientation space, and finally, deriving a closed, continuous implicit surface approximation for the quaternion orientation-space boundary whose interior represents the complete space of valid orientations. We then extend this formalism so that it can also handle coupled joints, which we treat as parent and child joints, such as the shoulder and elbow, or the hip and knee. We represent the space of valid configurations for the child joint as a function of the position of the parent joint, which is itself represented by a voxelisation of the quaternion field of allowable rotations. To build this representation from real experimental data, we have developed a motion capture protocol that relies on optical motion capture data acquired using the Vicon^{TM} system, which we have used to model the joint limits and their dependencies for the shoulder and elbow.

A major advantage of this quaternion representation is that it provides us with a rigorous distance measure between rotations, and thus supplies the most natural space in which to enforce joint-angle constraints by orthogonal projection onto the sub-space of valid orientations [Shoemake1985]. Furthermore, it is not subject to singularities such as the Gimbal lock of Euler angles or mapping rotations of 2π to zero rotations. As a result, it becomes easy to incorporate these sophisticated constraints into a motion tracking algorithm using standard constrained optimisation techniques [Baerlocher and Boulic1998]. We will demonstrate this by showing that this approach dramatically improves performance of an existing system [Plänkers and Fua2001] when attempting to track complex and ambiguous upper body motions from low quality stereo data. We chose to concentrate on shoulder and elbow motion tracking both because the shoulder is one of the most complex joints in the body and because the position of the arm constrains the elbow's range of motion. In short, the method we propose here advances the state-of-the-art because it provides a way to enforce joint limits on angular and axial rotation

of coupled joints while at the same time accounting for their dependencies. Our contribution in this thesis is therefore not the performance of the particular systems we happen to use for our experiments, but rather the framework we propose to enforce joint limits. It is generic and could be incorporated into any motion-tracking approach that relies on minimising an objective function.

1.5 Chapter organisation

Chapter 2 reviews the various motion capture systems that currently exist for acquiring human motion, and the methods that allow to recover the animation parameters on the basis of the collected data. Chapter 3 deals with the issue of human body modelling, briefly presenting existing models and their applications, then moving onto the body models we will be using for our specific motion capture purposes, and the reasons that motivated our choice. Chapter 4 handles the problem of state-space pruning, starting with an overview of existing constraints, on the one hand for data segmentation, and on the other hand, at the model-level. We will cover the constraints that we enforced in the case of 3D reconstruction and tracking, as well as the joint limits derived for the human upper arm, including the results obtained in both cases for motion capture, and in the latter case, for animation as well. In Chapter 5, we will reveal the secret of our joint limits determination technique, from data collection to joint limits representation, not omitting of course to present and illustrate related efforts in the computer graphics and vision, as well as medical communities. Some relevant implementation details will be given in Chapter 6, followed by our conclusion in Chapter 7, where we will remind the reader of the main strengths of the work involved, and also recognising its limitations and possible future extensions.

Chapter 2

Motion capture state-of-the-art

Motion capture being the centre-piece of the current work, we will, in the present chapter, give a brief overview of the various motion capture systems available on the market, these allowing mainly data acquisition. The process of converting the collected data into a usable animated human model is in general carried out independently from data acquisition, and numerous solutions have been proposed to handle the problem, with more or less success. The first section presents the various systems, and the advantages and drawbacks associated with each. The second section reviews the existing applications for recovering postures from specific input data.

2.1 Existing motion capture techniques

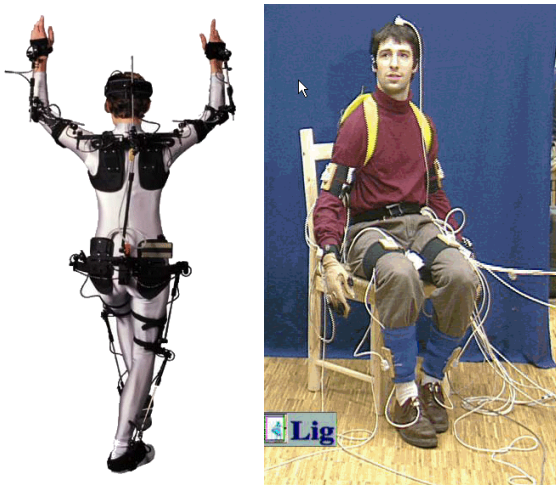
Motion capture can be defined as "the process of recording live motion and translating it into usable mathematical terms by tracking a number of key points in space over time, and combining them to obtain a single 3D representation of the performance." For our readers that are not so familiar with the terms involved, we would like to specify that by a "3D representation of the performance", we mean a 3D representation of the captured subject that for each image frame of the live motion adopts the correct pose with respect to the original motion. "Mathematical terms" are the various motion parameters that allow a 3D human body representation to adopt a certain posture, for example the angles at each joint of the body. By "tracking", we are referring to the notion of following the progression of a given item over time, i.e. over all the frames of a motion. More specifically, if we have located a certain object in a given frame, we need to be able to locate and identify the same object in the next frame and so on throughout the entire motion. Finally, by "key points", we refer to image features that will subsequently allow us to determine the position of the human body 3D representation. These key points can take many forms depending on the motion capture technique involved, but in all cases are landmarks on the human body (limb extremities, body contours, markers, etc.).

We will present the various motion capture systems, roughly in their order of appearance on the market in the context of computer animation, thus potentially allowing a trend in motion capture to emerge, which we will come back to later.

2.1.1 Mechanical motion capture, or "exo-skeletons"

An exo-skeleton consists of a suit composed of a group of metallic structures linked by accelerometers or similar devices located at the major joints, the suit being connected to an interface that allows the direct retrieving of the positional and rotational data of each piece.

A device of this kind is shown in Figure 2.1(a), and has obvious drawbacks: it is not only cumbersome but also restricts natural motion by its sheer presence, exerting a non-negligible force on the performer's joints. Also, the sampling rate for this kind of system is low and the system is inflexible as the skeleton configuration is fixed.



(a) (b)

Figure 2.1: (a) Exo-skeleton motion capture device; (b) Wired electro-magnetic suit.

Exo-skeletons don't exactly come cheap either, but for all this, they have the advantage of yielding immediate joint angle values within a capture space that is only limited by the length of the cables connecting the interface to the suit, therefore making it an ideal system for real-time motion capture.

2.1.2 Electro-magnetic motion capture

Older versions of electro-magnetic suits, such as the Ascension^{TM} "Flock of Birds" shown in Figure 2.1(b), consisted of a set of magnetic receivers placed on the joints of the subject, each one being connected to a computer via cables. A nearby magnetic transmitter allowed the measuring of the spatial relationship of the sensors, with immediate transferral to the computer. As can be readily understood from the picture, the suit was extremely cumbersome and the high risk of entanglement in the cables therefore condemned this device to be limited to a certain type of motions and to a restricted capture space. Nowadays, wireless electro-magnetic devices are available, such as Polhemus StarTrak^{TM}, thus solving the problem posed by the cabling.

Wireless or not, the system nevertheless experiences interference through the presence of any external magnetic fields or metallic objects, therefore restraining its field of use. It suffers from similar short-comings as the exo-skeleton, in the sense that the sensor configuration is either difficult to change, or the number of receivers limited.

To its credit, it remains a relatively low-cost solution, in the same range as the exo-skeletons, and the position of the sensors is obtained immediately, therefore making it a candidate for real-time motion capture. It should be noted that contrary to exo-skeletons, the rotational values of the joints are not retrieved and need to be computed. However, as each sensor is uniquely identified, it is just a question of matching the joint positions of the character to be animated with those of the sensors carried by the actor. One possibility for carrying out this task is to recursively compute the rigid transformations from one joint to another throughout the skeleton, once the virtual character has been scaled to the size of the actor and oriented in the same fashion, as in [Molet *et al.*1996] and [O'Brien *et al.*2000]. Another option is to use inverse kinematics techniques, where the sensor positions are the goals to attain, and the parameters of the virtual character's skeleton are computed using an optimisation method, as in [Bodenheimer *et al.*1997].

(a) (b)

Figure 2.2: (a) Optical motion capture markers on actor Andy Serkis during the shoot of "Lord of the Rings"; (b) Gollum, the character animated with the motion captured sessions.

2.1.3 Acoustic motion capture

Acoustic systems use high-frequency sound waves in order to determine the position of objects, thus requiring the presence of audio receivers placed around the capture area. Audio transmitters are placed on the joints of the subjects and transmit a sound at a regular interval. The position of each transmitter is computer by estimating its distance from the audio receivers using the speed of sound. The distances computed for each audio receiver are then triangulated in order to infer the position in 3D space of the sensor.

The advantage of such a system is that the cost of it remains low, but requires the same type of cabling as electro-magnetic devices. An additional drawback is that the sensors need to be within the line of transmission of the three audio receivers at all times, in order for a 3D position to be inferred, and therefore complicated movements will cause the loss of positional data. Furthermore, the data for all transmitters cannot be computed at the same time, forcing the actor to not move until the last transmitter has emitted a sound, which is not exactly convenient for fast motion, or for any motion, as a matter of fact. Also, once the 3D positions of all transmitters recovered, joint angle values still need to be computed, this requiring the same computational effort as for the electro-magnetic system. Needless to say that the technology by definition suffers from possible interference from other sound sources.

2.1.4 Optical motion capture

Optical motion capture systems function in a similar way to acoustic systems, except that light is used instead of sound for inferring sensor positions.

In such a system, a certain number of calibrated infrared cameras is placed around the scene (typically six to eight) and the subject wears reflective markers. Given the intrinsic parameters of each camera and given the position of the markers in each camera view, the position of the sensors in 3D space can be computed using stereo triangulation. As the name indicates, a minimum of three cameras is necessary in order to reliably infer a 3D position[1], as will be explained in the Appendix.

[1] Two cameras are in theory sufficient, but can cause depth ambiguity problem (see the section on Stereovision in the Appendix).

The advantage of such a system is that the marker configuration is totally flexible, as is the number of markers that can be used. Reflective markers are not cumbersome to carry and do not constrain the performed motion in general. Furthermore, only the focal length of the cameras determines the size of the capture area, which can therefore be extended or reduced at will, and the sampling rate is very high, therefore allowing the capture of fast and complex motion, if necessary. The precision that can be obtained makes it a valuable tool not only in the computer graphics community, but also in the medical area where it finds the largest part of its customers [Menache1999].

The drawbacks are similar to those of the acoustic system, in the sense that markers need to be visible in order for their position to be inferred, but thanks to the large number of cameras, the chances of a marker being seen by at least three of them is high. However, nothing can save the system from being sensitive to other light sources, this more often than not causing the presence of "ghost markers" in the 3D reconstruction. Also, such systems are prohibitively expensive, ranging from 100'000 to 300'000$ or more, and in terms of human intervention, the need for post-processing remains very high. As in the case of an acoustic system, the joint angle values need to be inferred from the 3D marker positions. However, contrary to the other systems mentioned, the reconstructed markers are anonymous and therefore need to be identified before any posture recovery is possible. Many commercial motion capture systems include such a tracking software package, where each reconstructed marker is identified manually in the first frame, and then tracked over the sequence, thus greatly reducing the overhead of the pose recovery process. However, the tracked sequence needs to be visualised by an operator and the labelling errors manually corrected, as there is so far no such thing as an error-free system. Furthermore, a *modus operandi* needs to be defined for markers that are occluded over a certain number of frames. Position interpolation schemes are generally available in commercial packages but unless the occlusion time frame is very low, such a solution does not yield good enough results.

Optical motion capture systems are presently the most popular means of capturing or analysing motion as it for the moments is the technology that provides the most accurate data. It has been applied with great success to a great variety of well-known video game or movie characters, such as the recent Gollum in Peter Jackson's "Lord of the Rings" (Figure 2.2).

2.1.5 Motion capture from video

The rising interest of the scientific research community for motion capture from video is not only due to the fact that it presents innumerable difficulties, has low requirements in terms of hardware, but also that it keeps researchers gainfully employed (government officials take notice, here is the solution for cutting unemployment rates). If the motion capture from video problem could be successfully tackled, it would be the perfect solution. There would be no need for anyone to sport any fancy devices or markers, be subject to various system-related constraints or invest a hair-raising amount of money. All that would be needed is one or several video cameras in terms of system setup, and the show would be on.

Now here comes the catch: the problem of motion capture from video has been actively worked over a decade, but the generic solution is still at large. To this day, it has not fared better than its peers because a lot of limiting conditions need to be imposed for it to yield a satisfactory result.

Obviously, this kind of motion capture presents the highest interest to the computer vision community, as it transfers the bulk of the work from the electronic gadgets to human ingeniousness. Data collection is easy and cheap, but the amount of post-processing involved is, to this day, very consequent. For an overview of the work carried out in this area, we refer the reader to the excellent surveys and reviews published by [Aggarwal and Cai1999], [Gavrila1999] and [Moeslund and Granum2001]. A more up-to-date review can be found in [Moeslund2003]. In the two latter review papers, the surveys are extended to post-processing methods also for optical motion capture, as this also qualifies as camera-based motion capture. Furthermore, in the area of scientific research, optical motion capture is not an issue that is considered solved, in spite of the fact that the commercial community claims to deliver 100% reliable motion capture packages. We will therefore, and from here on, focus on these two motion capture techniques and their possible improvements through computer vision and graphics methods.

For the record, the general field of motion analysis in computer vision includes a broad range of problems, such as:

- **motion recognition**: low-level features of interest are retrieved and used to describe the human movement. They are then compared to learned and/or classified features, in order to determine to which motion type they belong, thus identifying the motion.

- **motion tracking**: where the aim is detection of a presence and localisation from one frame to the next, but where it is not necessary to extract all motion information, as feature points will often do the trick.

- **motion capture**: where one wishes to recover the full body pose in order to apply it to another model or analyse the data. In this case, it is an established fact that a subject is present in the scene, and its whereabouts have been pin-pointed, but its exact posture needs to be determined.

2.2 Various approaches for estimating motion parameters

In this thesis, we will be dealing mainly with the problem of motion capture in terms of posture recovery, unless tracking and pose recovery are handled simultaneously, if they are inter-dependent. The reason why camera-based motion capture is still an on-going research topic was explained in the previous chapter, where we also laid out the difficulties that any person tackling this problem will inevitably have to face. The various methods that have been published to this day are what we will detail in the present section.

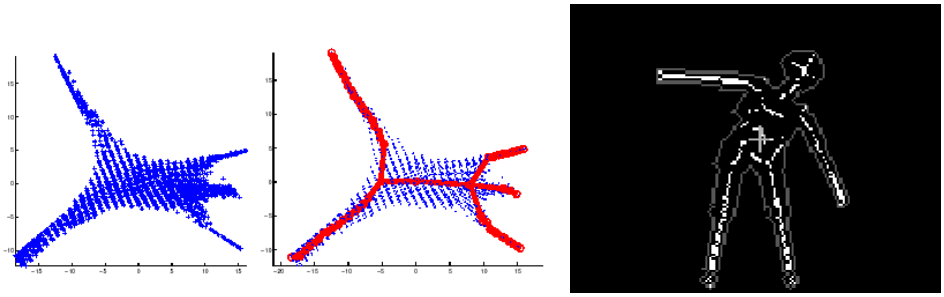
In the afore-mentioned state-of-the-art papers, the existing methods are classified either according to the means used to solve each sub-problem, or by distinguishing between 2D and 3D methods, as well as methods using a body model and those that do not.

With 2D or 3D methods, it is the dimensionality of the recovered motion that is being referred to. A certain number of papers deal with recovering 2D motion parameters and applying them, for example, to cartoon characters, or analysing them in terms of motion in one plane only. However, in the perspective of general motion capture applications, such as those listed in Section 1.2, we will be considering here only three-dimensional motion.

When it comes to recovering 3D postures, nearly all methods use a pre-defined model representing the human body with more or less precision. In a few rare cases, a human body representation is derived directly from the segmented 3D data, such as those that make use of volumetric data [Chu *et al.*2003, Iwasawa1997]. Chu et al. transform a set of 3D points representing the human body volume into a pose-invariant intrinsic space posture using the isomap transform [Tenenbaum *et al.*2000], extracting the principle curves and subsequently back-projecting them into Euclidean space to yield the corresponding skeleton posture (Figure 2.3(a)). Iwasawa applies the simpler method of extracting representative human body points from the silhouettes using three cameras, the skeleton joint positions and rotations being inferred using genetic algorithms (Figure 2.3(b)) on the basis of learned data.

In the more general case, however, when it comes to recovering pose, the use of a human body model is inevitable as it presents a two-fold advantage. On the one hand, the pose of the model can been recovered from the data using inverse kinematics-like techniques, the corresponding joint parameters following immediately from this optimisation step. On the other hand, the model providing knowledge of the kinematic functioning of the body and its shape properties, it presents a useful means of either pruning the solution space, or predicting and/or disambiguating multiple hypotheses that arise especially in the case of self- occlusion and collision.

Pose recovery using a model is also referred to as “analysis-by-synthesis”, where the model is used to predict the posture in the next frame, the features of the synthetic model in the predicted posture being matched to the data in order to determine the correct pose. The following state-of-the-art groups papers by the method used for pose recovery.



(a) (b)

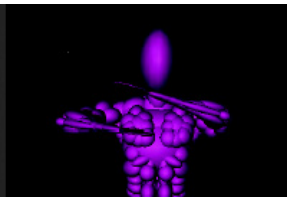
Figure 2.3: Skeleton figure derivation: (a) from the isomap transform [Chu *et al.*2003]; (b) from silhouettes [Iwasawa1997].

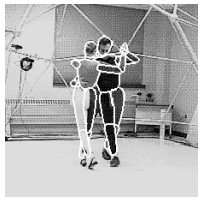
Divide-and-conquer techniques [O'Rourke and Badler1980] were the first to use a model-based approach and pose recovery was carried out one body part at a time, following links from one body part to the next until the motion parameters of each have been successfully determined. The 3D model is used to predict a posture in state-space, and by projecting it onto the image, it allows the defining of regions where the body extremities should be located. Segmentation of the body parts then enables the system to determine the joint positions and angular values, by simulating motion of the synthetic model until the "best-guess" position is found. [Holt *et al.*1997] followed a similar approach, except that the motion of each body part is assessed on the basis of several image views taken from different perspectives. [Kakadiaris *et al.*1994] used a physics-based approach, first introduced in earlier work by [Metaxas and Terzopoulos1993], with forces acting on the various body parts of a customised model in order to align them with the silhouette extracted from the image, using a Kalman-filter based estimator to predict model parameters. They later refined this method for a multi-view approach and continuous surface models, as described in [Kakadiaris and Metaxas1996, Kakadiaris and Metaxas2000], and illustrated in Figure 2.4(a). More work using the same basic technique was published by [Kameda *et al.*1993] using silhouettes and monocular sequences; [Munkelt *et al.*1998] who matched coloured markers in stereo images; [Zheng and Suezaki1998] who worked on monocular images, using manually-set keyframe postures, then performing interpolation between the keyframes and image-to-model shape correlation to determine the best match.

Inverse kinematics-like techniques An extremely popular approach is to use a method similar to the one commonly employed in robotics, where the various joints in a robotic structure are modified until the end effector reaches a given goal position, using inverse kinematics to determine the joint angles that satisfy the condition. For the purpose of pose recovery from image data, a parameterised model of the human is designed as a hierarchical structure, each link having the rotational properties that correspond more or less to the real-life situation. The extracted features act as the goal to be fulfilled, and inverse kinematics techniques are used to determine the best posture.

Gradient-descent methods A certain number of such techniques use gradient descent optimisation methods in order to determine the posture in monocular sequences. This is the case in [Goncalves *et al.*1995] and [Rehg and Kanade1995], where a parameterised model of, respectively, the human arm and hand, is matched to contours, applying Kalman filtering for pose prediction, and in [Zhuang *et al.*1999] where the same notions were applied to full body posture recovery. [Yamamoto and Koshikawa1991] also worked on arm posture recovery, but using the optical flow of several points of interest on each body part to determine the motion of the point, thus retrieving the motion of the entire body part using gradient descent. [Bottino *et al.*1998] used multiple cameras placed around the scene to infer a voxelised volume of the captured subject, then carrying out gradient descent on their model to fit the extracted volume. [Bregler and Malik1998a, Bregler and Malik1998b]

(a)





(b) (c)

Figure 2.4: Posture recovery from video sequences: (a) Upper limb physics-based tracking [Kakadiaris and Metaxas2000]; (b) Pose recovery from stereo data [Plänkers and Fua2001]; (c) the tracking problem as a best pose search by model synthesis [Gavrila and Davis1995].

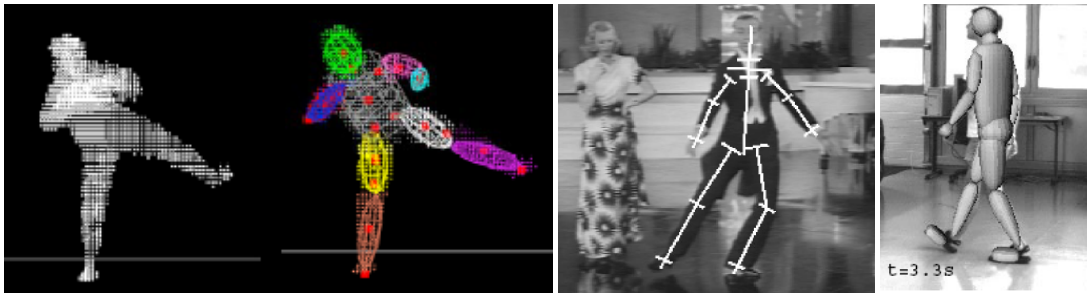
proposed to express model rotation using exponential maps and taking into account the change in segment lengths depending on the view perspective. In this manner, they obtained a set of linear equations that can be solved using differential motion estimation, for matching the model body parts to image regions extracted using image intensity. [D'Apuzzo *et al.*2000] and [Plänkers and Fua2001] recover the posture of a user-tuned body model, whose shape is recovered on-the-fly, and match the model to stereo data reconstructed from three cameras and the silhouettes in each view (see Figure 2.4(b)). [Demirdjian2003] also uses 3D data extracted from stereo images to fit a mesh-based model to the data, using Iterative Closest Point optimisation. More work in this direction includes [Wachter and Nagel1999], for monocular posture recovery of walking persons using edge detection; [Yonemoto *et al.*2000] who detect the main body parts by colour matching in multiple views and then estimate posture in real-time using such an inverse kinematics technique; [Ringer and Lasenby2000] who determine the 3D pose of human legs on the basis of optical markers captured in a multi-view environment; [DiFranco *et al.*2001] determine 3D body posture using gradient descent not only on the matched features, but also directly on the constraints, their method nevertheless requiring that image-to-model correspondences be defined manually, as well as a certain number of keyframes; [Rehg and Kanade1994] track and recover the pose of a human hand model in real-time from grey-scaled images by matching boundary points. More recently, [Carranza *et al.*2003] applied non-linear optimisation for recovering the posture of a model from silhouettes extracted from multiple views. Optimisation is, however, carried out independently for each body part, in order to avoid falling into local minima.

Search methods Other methods rely on synthesising model postures and define posture recovery as a search problem aiming at finding the joint parameters of the model that best correspond to the extracted features. This is the case in the work of [Gavrila and Davis1995], who, in a multi-view framework, used their model for pose prediction. The similarity measure between the model and image contours is explained by Chamfer distances between point sets, where the motion parameters of each body part are perturbed and the "best-first" solution retained for each (see Figure 2.4(c)). [Kuch and Huang1995] used a similar approach for hand pose recovery from a single view, matching the model with the extracted foreground of the image using a defined

distance measure, as did [Heap and Hogg1996], as well as [Lee and Kunii1993]. [Perales and Torres1994] extended the method for matching of a full body model either in an interactive or automatic manner. In all three cases, gradient descent is performed in order to find the optimal solution within the set of proposed synthesised postures for each frame. [Wilhelms *et al.*2000] used active contours corresponding to the projected silhouettes, which have anchor points to the body part segments, each one being separately adjusted to align the model contour with the silhouette. The posture is interactively set for a number of keyframes, to aid the posture recovery process. [Allen *et al.*2003] combine the recovery of shape and motion parameters, and for the latter, they employ body landmarks to which skeleton joints are fitted using inverse kinematics and searching for the posture that yields the best match with the data.

Probabilistic techniques Determining the best matching posture can also be carried out using probabilistic techniques, as was done by [Hunter *et al.*1997] for matching an arm model to contours in a single image, using the Expectation-Maximisation (EM) algorithm. The E-step predicts the posture of the model and the M-step verifies the hypothesis and the probability distribution is updated according to the observed correspondence between image and model features. [Choo and Fleet2001] use hybrid Monte Carlo filtering to estimate the postures of the model of the image sequence, using markers positions representing joint locations as matching features. [Segawa *et al.*2000] used constraints directly applied to EM-based posture estimation, detecting coloured body parts in monocular sequences. [Mikic *et al.*2003] used a Kalman filter to estimate model parameters on the basis of voxels of the human subject from multiple calibrated cameras, [Ude and Riley1999] also using Kalman filtering, match and track body parts via optical flow in a two-cameras environment. As to [Cerveri *et al.*2003], they applied the Kalman filtering technique to estimating posture on the basis of optical motion capture data. [Drummond and Cipolla2001] used Gaussian and Laplacian density distributions representing the probability that a given joint posture yields the observed edges in the image. These statistics are propagated along the kinematic chain, least-squares minimisation allowing the determination of the best solution. The technique was applied to monocular sequences for non-human articulated structure, but image data from at least another view had to be used in the case of a human body to aid the tracking process, as in both cases the process is real-time.

Multiple hypotheses techniques [Cham and Rehg1998] use multiple hypothesis tracking (MHT) that generates multiple hypotheses using a set of Kalman filters, the likelihood of each hypothesis being determined by templates at the pixel level. MHT methods take into account the fact that there is no unique configuration explaining the observed image data. Furthermore, in this case, a 3D and a 2D model are simultaneously maintained, the 2D model for tracking in the images, and the 3D model for reflecting the posture in the scene. Each model is projected into the space of the other one, in order to enforce consistency between the inferred 2D and 3D postures, this being the so-called Scaled Prismatic Model (SPM) introduced by [Morris and Rehg1998]. [Davison *et al.*2001] and [Deutscher *et al.*2000] also use MHT, multiple solutions being maintained at each time step, on the basis that configurations with a smaller likelihood should not be thrown away immediately, the likelihood of each sequence of solutions being determined over a large frame window. They use the particle filter known as Condensation, introduced by [Isard and Blake1998], which allows the propagation of multiple hypotheses. [Sminchisescu and Triggs2003] have obtained some of the best body pose recovery results from monocular sequences to date, by astutely mixing MHT and global optimisation of a cost function that is defined locally as a probability distribution. The cost function is estimated on the basis of multiple image features, in order to reduce the chances of falling into local minima, and is defined as an error distribution of the mapping of the model onto the image, the uncertainty measure of this matching being computed using the covariance matrix. Moving along the directions of maximal uncertainty, new samples are obtained thus generating a new set of hypotheses propagated by the MHT. The technique was later on extended to a multi-view framework, where silhouettes from all views were extracted. These were used to determine the approximate 3D position of the model, and then served as matching features for the model contour [Sminchisescu and Telea2002]. [Ringer and Lasenby2002a] also maintained multiple tracking hypothesis for the segmentation of optical motion capture markers, using a Bayesian approach for the tracking itself [Ringer and Lasenby2002b].



(a) (b) (c)

Figure 2.5: Posture recovery from video sequences: (a) Pose recovery from volumetric data [Mikic *et al.*2003]; (b) Multiple Hypothesis Tracking [Cham and Rehg1998]; (c) multi-view MHT [Sminchisescu and Telea2002].

Measurement equation techniques Some researchers have endeavoured to recover postures using precise measurement equations, relying on the segmentation of images and detection of landmark features for directly inferring the entire set of model parameters. An example can be found in [Noma *et al.*1999], who construct transition graphs to recognise a small set of motion types, each motion type defining constraints that allow for the motion parameters to be inferred on the basis of retrieved body part regions tracked using colour distributions in monocular sequences. Similarly, [Ren and Xu2001] estimate the posture of an arm model from stereo data, where the detected hands and eyes serve as initially extracted features for determining the pose from various parameter equations. [Moeslund *et al.*2003] first determined such equations by studying the parameters of the shoulder joint, constructing graphs and a look-up table from the observed data. This allowed the recovery of the motion of an arm model on the basis of the segmented hand region, and subsequent matching of the model contours to the image-extracted silhouette. The collected data was used to construct an anatomical shoulder model used for estimating pose from silhouettes in a single view set-up [Moeslund *et al.*2002] . [Chua *et al.*2002] have established a set of equations explaining the joint parameters of a human hand, and use extracted contours and feature points to infer all rotational values. [Cheung *et al.*2003] apply such motion constraints between body parts to a full body model matched to silhouettes from multiple camera views. [Liu *et al.*1999] track reflective markers in a video sequence, using a motion library providing motion-specific model parameters that are adjusted to comply with the known perspective camera projection. [Theobalt *et al.*2002] simply use forward kinematics computations to position a body model within the visual hull, resulting from the combination of body part bounding boxes from each camera view. In order to do this, body part extremities are segmented and their exact positions located in each frame, therefore allowing the immediate positioning of a good part of the body joints, furthermore assuming that global motion direction is invariant.

Learning techniques Finally, the last category of posture recovery methods consists of those relying on learned motion, which is the case of [Zhao *et al.*2002]. The full body model posture is estimated by clustering and coding the learned states and using the Minimum Length Descriptor to retrieve the best fit, based on texture and boundary matching. [Ong and Gong1999] submitted their upper body model to a learning phase, then inferring postures from silhouettes using the Condensation algorithm for tracking, in single or multiple camera views. [Rosales and Sclaroff2000] learned the combined silhouettes corresponding to model configurations, defined in terms of optical motion capture marker positions, and used this information for recovering body postures in a single view. The model joint parameters are not directly inferred, but could be, using a minimisation technique. [Wu and Huang1999] learned the set of possible poses of a hand model, and recovered hand posture in an image by supposing that tracking and segmentation have already been carried out. Global hand motion is determined using an optimisation technique and then finding the hand posture by applying a genetic algorithm.

[Lin *et al.*2001], making the same assumptions, likewise learned their hand postures from motions performed using a CyberGlove, applying Principal Component Analysis to store their motion parameters in a more compact way. [Dockstader and Tekalp2002] constructed probabilistic distributions for human walking from multiple views to aid the matching of the model to the extracted visual hull.

Chapter 3

Body models

A complete motion capture process starts with the definition of the body model to be used for analysis-by-synthesis. More precisely, the body model will allow the synthesis of postures whose features are subsequently compared to the data features, whatever their nature may be. This comparison tells us how far away we are from the solution, and eventually gives a hint of the direction in which to move to improve it.

Such a model therefore intrinsically represents the space of all possible solutions, and hence it is paramount that it can synthesise postures explaining the observed data features, but also that its own defining features can be related to the features extracted from the image data. As posture recovery is also a process that should be, if need be, carried out relatively fast, it is also in our best interest that the model not be represented by an excessive number of parameters that would cause needless computation.

To summarise, when defining a body model for motion capture purposes, one should heed the following:

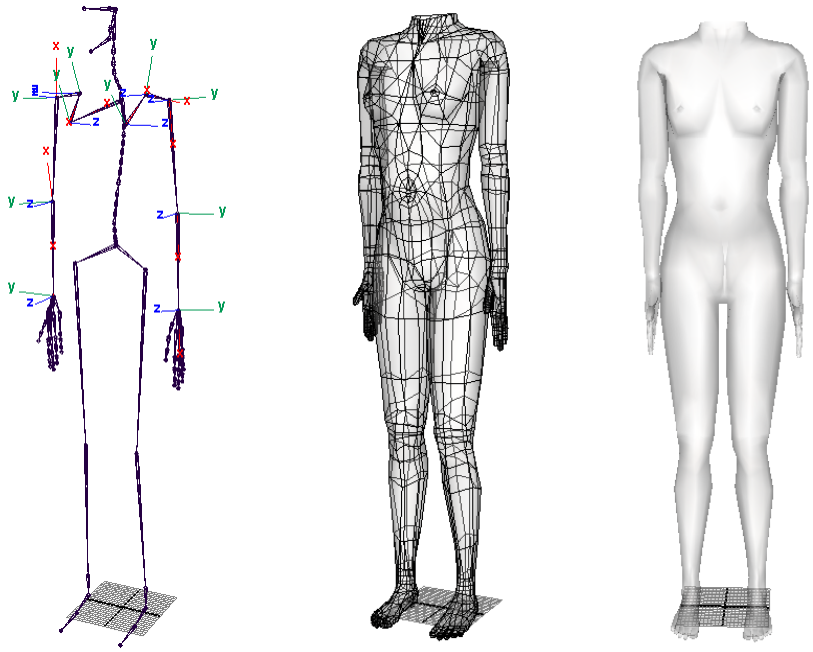
- The number of parameters defining the model's configuration should be sufficient to recover the animation with the wanted precision, but without adding unnecessary complexity.

- One should, likewise, be able to put in correspondence the elements defining the shape of the model with the features representing the human body shape in the image data, without adding unnecessary shape details that do not contribute to pose recovery.

3.1 Human body representations for motion capture

The body models used for providing *a priori* motion knowledge to the motion capture process vary in complexity depending on the application. A model typically includes a skeleton representation, consisting of bones and joints, and optionally a shape layer defining the body outline. In very rare cases [Kakadiaris and Metaxas1995], the body model consists only of the shape layer, where each body part is modelled separately. The translation and rotation of each body part is computed during posture estimation, instead of the rotational values at the skeleton joint level.

The complexity of a body model is not only defined by the number of model layers, but also by the intrinsic complexity of each layer itself. An example of a skeleton-only and a skeleton-and-shape model is shown in Figure 3.1. The representation and parameterisation possibilities of each are detailed in the following sub-sections.



(a) (b) (c)

Figure 3.1: Complete body model: (a) skeleton layer; (b) shape layer and (c) rendered body with skin.

3.1.1 Skeleton models or "stick figures"

The skeleton (Figure 3.1(a)) is, for obvious reasons, the required item in a body model, as it defines the pose and shape variations at the higher levels[1].

The skeleton is generally represented by segments linked by nodes that have one, two or three rotational degrees of freedom (DOFs), depending on the joint. The segments represent the bones and can vary in length, expressed by a translation parameter. Their thickness is not taken into account, as this would unnecessarily complicate the model, given that a soft tissue layer can be added to represent shape, as will be discussed.

A joint between segments can be represented by any of the existing rotation formalism, such as Euler angles, axis-angles, exponential maps or quaternions. Their respective properties are described in the Appendix.

Each joint in the skeleton has a local co-ordinate system associated to it, and the rotation at the joint is defined with respect to the three orthogonal axes. In general, and as is the case in Figure 3.1(a), one of the axes is always aligned with the segment whose orientation is determined by the joint rotation. In this general case, a 3D rotation can be defined in terms of angular and axial rotation, angular rotation defining the orientation of a segment in 3D space, and axial rotation performing a rotation of the segment around its aligned axis. Re-using the terminology of [Baerlocher2001], we will also refer to angular rotation as "swing" and axial rotation as "twist".

The various joint types in the human body have been classified by [Hamill and Knutzen1995], as illustrated by Figure 3.2. The pure rotational joints, that is, those not involving any bone translation, are:

- one DOF joints: hinge (one swing component, such as the ulno-humeral joint, i.e. flexion of the elbow) or pivot (twist component, such as the ulno-radial joint, i.e. twisting of the elbow)

[1] Note that in computer graphics models, it is the skeleton that drives the deformation of the soft tissues whereas in reality, the reverse occurs. Indeed, in real life, it is the muscles that allow the motion of the skeleton.

(a) (b) (c) (d) (e)

Figure 3.2: Various joint types: (a) the hinge joint performs rotation in one plane only; (b) the pivot joint allows only axial rotation, (c) the ellipsoidal joint rotates in two planes (no rotation around the axis itself); (d) the saddle joint allows rotation in two planes and a small amount of axial rotation; (e) the spherical or "ball-and-socket" joint rotates in three planes [BBC2003].

- two DOF joints: condyloid (one swing component and twist, such as the knee) or ellipsoidal (two swing components, such as the wrist)
- three DOF joints: saddle (as the thumb) and ball-and-socket (two swing components and twist, such as the shoulder and hip joints).

Representing 1 DOF or 2 DOF rotation presents no particular problem, and it can be expressed by a Euler angle around one or two axes. The axes around which no rotation is allowed are said to be "fixed".

The 3 DOF rotation can be represented either by quaternions, axis-angles, exponential maps or Euler angles, the problem being that the three latter formalisms come hand-in-hand with singularities. These are relatively easily avoided in the case of the axis-angle or exponential map, but not so in the case of Euler angles. To top it all, these also are seriously dependent on the chosen order of rotation around the axes. These issues are covered in the Appendix, and for a discussion on the representation of 2 and 3 DOF joints by these rotation paradigms, see [Grassia1998]. Quaternions, on the other hand, are void of singularities but have the double disadvantage of involving four parameters to represent a 3D rotation and having an unfortunate reputation of being mathematically more complex. Despite the increasing amount of incriminating evidence against them, Euler angles are still favoured by a great majority of human body models. The most likely explanation can be provided by the combined popular axioms that "Man is a creature of habit" and "if it works, don't fix it".

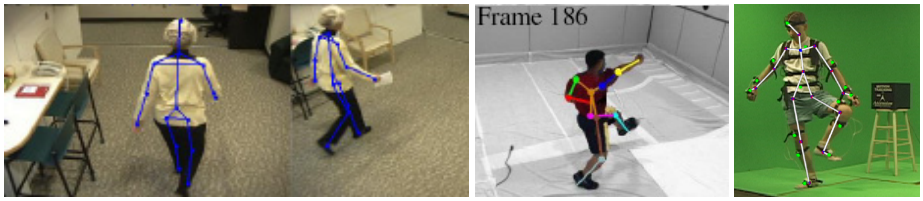
In any case, the degree of complexity of the skeleton layer is determined by the number of joints and segments that it includes, as well as the number of rotational DOFs allowed at each joint.

Such a stick figure will be preferred when one or several of the following are true:

- the significant features extracted from the image data correspond to joint locations;
- a shape layer does not provide sufficient added-value in order to justify handling this additional set of parameters;
- computation time is a major issue, such as in real-time applications;
- the precision of the recovered poses needs to be extremely high.

Some examples of motion capture applications using a stick figure model are shown in Figure 3.3, where a stick figure is respectively matched to a 3D visual hull extracted from image data, to silhouettes in multiple views, and to 3D joint positions acquired using an electro-magnetic motion capture device.

The degree of complexity of the stick figure is similarly constrained by the application, namely because:



(a) (b) (c)

Figure 3.3: Motion capture using a stick figure model: (a) posture recovery based on the extracted visual hull [Dockstader and Tekalp2002]; (b) silhouette-based tracking [Cheung *et al.*2003]; (c) posture determination on the basis of joint locations provided by an electro-magnetic motion capture device [O'Brien *et al.*2000].

- The method chosen for posture recovery might impose a maximum number of parameters of the model to make the computation method tractable.
- The lack of detail of the data features might render a certain number of DOFs useless in the sense that they could never be determined on the basis of the input data alone.
- In the case of character animation, the number of rotational joints and their respective DOFs at the model level should be at least equal to those of the target character. If this condition is not fulfilled, one runs the risk of obtaining unrealistic deformations at the higher body layers.

Hence, depending on various issues, a skeleton model can range from being a gross approximation of the mechanical functioning of a real human body, to a model where the joint structure and mobility correspond to biological reality, in which case the model is said to be biomechanical.

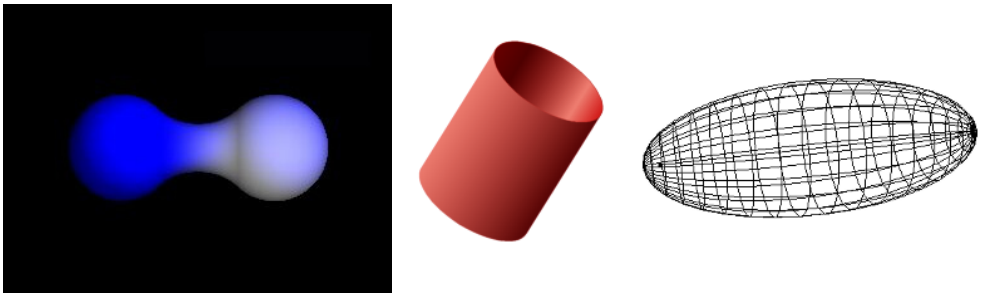
3.1.2 Body shape representations

The modelling of the human body shape (Figure 3.1(b)) can be carried out either by volumetric primitives, such as cylinders, metaballs or super quadrics, or by surface models, such as polygonal meshes.

Precision is usually proportional to the number of parameters used to represent it. Better precision can be achieved by increasing either the number of primitives in the case of volumetric representations, or the level of detail, in the case of meshes. However, high realism is usually not necessary in motion capture applications as the purpose of the body model is mainly to segment the input data.

Shape representation by meshes has, until recently, rarely been used in the case of motion capture as it is costly, due to the fact that it is an explicit representation, as opposed to the volumetric primitives that have an implicit mathematical formulation. Furthermore, their deformation is non-trivial and if local deformations are to be possible, a high number of control points is necessary, making this representation even more expensive. [Heap and Hogg1996] used a surface mesh (Figure 3.5(a)), but did not do so directly for posture recovery, but for learning the relationship between surface mesh configurations and the underlying hand posture, the various mesh states being represented as Point Distribution Models. These points correspond to a sampling around mesh vertices, dimensionality reduction being carried out using Principal Components Analysis. [Carranza *et al.*2003], on the other hand, actually use a mesh-based body model for recovering motion, as does [Demirdjian2003], such a representation becoming currently more common, due to increasing available computing power, and the fact that mesh models simultaneously provide motion and deformation parameters.

Super-quadrics [Barr1981] or simple geometrical primitives (spheres, cylinders, boxes) are often sufficient to provide a shape approximation of the human body. Super-quadrics (3.4(b)) are a crossing of quadric surfaces and solids, yielding shapes defined by an implicit equation whose parameters represent the size of the shape in three directions, as well as define a smoothly changing family of shapes from rounded to square [Leonardis *et al.*1997].



(a) (b)

Figure 3.4: Volumetric primitives: (a) metaballs; (b) cylinders and ellipsoids, both being special cases of the generalised super-ellipsoid.

Both super-quadrics (Figure 3.5(d)) and geometrical primitives (Figure 3.5(b) and (c)) define a shape layer that is rigid, and are used when the motion capture data is too noisy to be associated with precision to the contours of the body model, or when shape recovery is not part of the process.

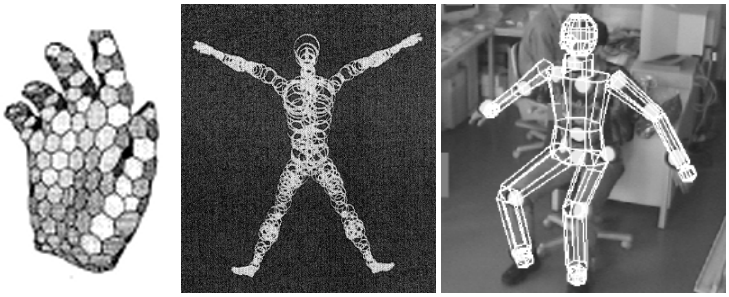
Metaballs, or "blobs", are commonly used in computer graphics, and are defined by a field function, as well as parameters that determine their blending properties (see Figure 3.4(a)) and their radius. The fact that they readily blend into each other is a very desirable property for representing shape, as the resulting surface is extremely smooth. They therefore present an ideal solution for motion capture applications that aim at recovering not only body posture but also body shape for a particular person (Figure 3.5(e)). Furthermore, because the metaball representation involves only a small number of parameters, it yields a smooth and realistic body shape while being computationally light-weight. It should also be noted that the exponential or linear potential field expression of a metaball has the effect of naturally limiting the impact of distant data features, thereby ensuring that noise or outliers do not affect too greatly the matching process. For a more in-depth explanation of the mathematical properties of the metaball representation, we refer the reader to Section 5.3.

3.2 Models used in this work

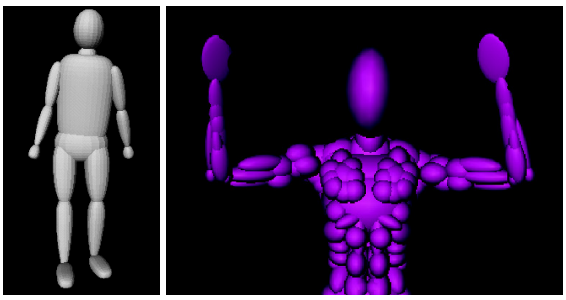
The human joint models used for animation and/or tracking are naturally simplified version of the actual anatomic joints, as computation time and the number of parameters would become prohibitive in the cases, respectively, of animation and tracking. However, the increasing computing power that we have at our disposal allows more sophisticated models to be used even for computer vision applications.

As pointed out in [Plänkers2001], the relevance of a more realistic model for tracking should not be overlooked. Attempting to fit a too simplistic a model to the data captured from a real human being will in some cases fail to produce a correct posture, as no configuration of the model really matches the data. In the light of the considerations laid out in Section 3.1.2, one should opt for the best trade-off in terms of body model detail, keeping the number of parameters low enough so as not to under-constrain the problem, but nevertheless precise enough to recover the desired motion details.

Depending on the quality of the 3D data, the chosen model can always be simplified, if need be, but to obtain the highest possible quality in terms of posture recovery, we will from the start define a relatively complete human body model. The structure of the skeleton layer will be compliant with the H-Anim standard, for maximum re-usability. H-Anim was defined by the Humanoid Animation Working Group to be the standard way of representing humanoids in VRML (Virtual Reality Markup Language). The aim of such a standard was to create a library of inter-changeable humanoids, where humanoids created using the software of a given author could readily be animated using an application by another author. The standard and resulting model skeleton are shown



(a) (b) (c)



(d) (e)

Figure 3.5: Shape representation models in motion capture: (a) polygonal mesh for representing a hand model [Heap and Hogg1996]; (b) over-lapping spheres, in the case of [O'Rourke and Badler1980]; (c) various geometrical primitives, depending on the limb, for [Munkelt *et al.*1998]; (d) super-quadrics, probably the most popular approach to date, [Sminchisescu and Triggs2003]; (e) metaball-based model for matching to stereo data [Plänkers and Fua2001].

in Figure 3.6.

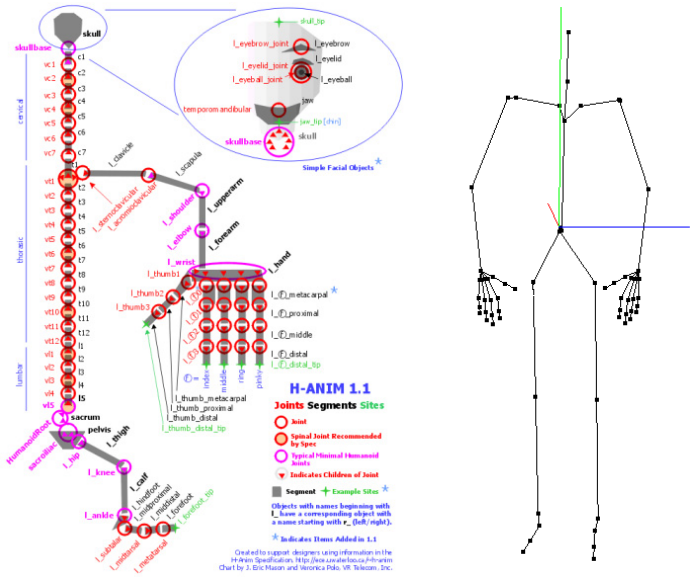
As we have mentioned in the previous chapter, the findings of this thesis are to be illustrated on the human upper limb, as many before us have chosen to do, our motivation for this being that the shoulder complex is acknowledged to be the most complicated joint structure in the human body. The various considerations and methods laid out in the chapters of this document should therefore be easily applicable to any other human body joint, as all joints in the human body can be viewed either as analogous to - or as a simplification of - the joints at the shoulder complex and elbow levels. The motion capture process will however always be carried out on a full body model such as the one in Figure 3.6.

We will be taking a fine-to-coarse approach of upper limb joint geometry given that the required complexity is application-dependent, starting with the definition of some shoulder-related terminology and a quick overview of its anatomical structure.

3.2.1 Upper limb motion terminology

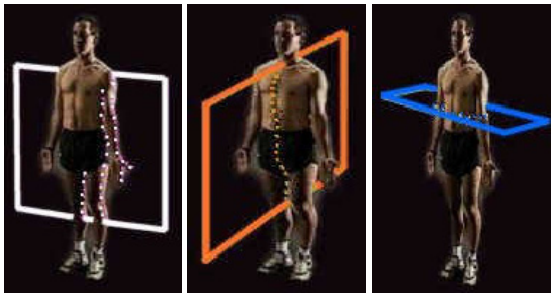
In general terms, one can refer to the motion components in terms of swing (angular rotation) and twist (axial rotation). By angular rotation is meant the motion of a body about a fixed point or axis, this being equal to a plane passed over at the point or axis by a line drawn to the body. Axial rotation occurs when a body is rotated about its own principle axis.

In the specialised literature dealing with human motion, one refers to:



(a) (b)

Figure 3.6: (a) H-Anim standard for humanoid articulated structures; (b) stick figure body model.



(a) (b) (c)

Figure 3.7: The motion planes: (a) coronal; (b) sagittal; (c) and transversal plane. All pictures are courtesy of [Drill1992].

- abduction and adduction as motion in the coronal plane;
- flexion and extension as motion in the sagittal plane;
- rotation as motion along the limb axis.

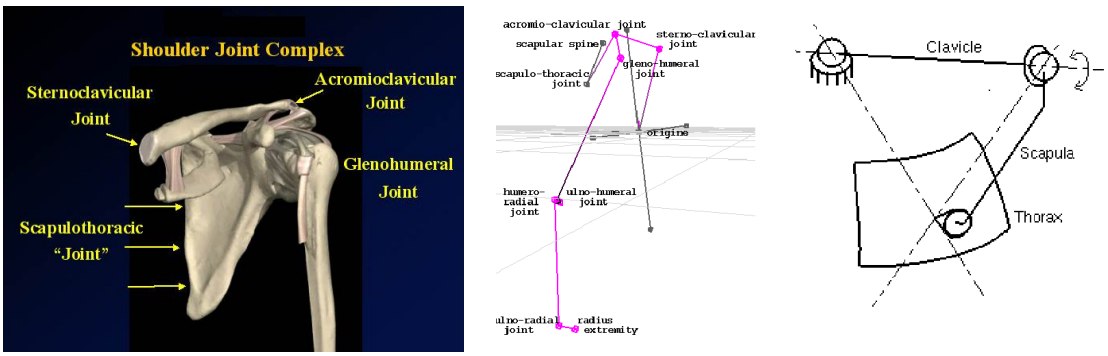
An illustration of the various rotation planes is shown in Figure 3.7.

In most areas dealing with human motion, computer graphics and vision set aside, the initial (or 'resting') pose of the arm is alongside the body, palm facing inwards, fingers pointing to the ground. All motion is therefore measured with respect to this initial pose. In Figure 3.8, we illustrate these concepts, using the example of upper arm motion.

In our case, the definitions of abduction/adduction and axial rotation (referred to as 'twist' from here on) will remain compatible with the medical terminology. However, in the case of flexion/extension, we will from here

(a) (b) (c)

Figure 3.8: The motion components of the shoulder: (a) abduction and adduction, respectively away and towards the mid-line; (b) flexion and extension, respectively away and towards the mid-line; (c) inner and outer rotation, respectively downwards and upwards. All pictures are courtesy of [Drill1992].



(a) (b) (c)

Figure 3.9: Shoulder joints: (a) 3D model of the shoulder compound [University of Iowa1992]; (b) anatomical articulated structure of the upper arm; (c) schematic drawing of the inter-joint relationships of the shoulder complex [Maurel1998].

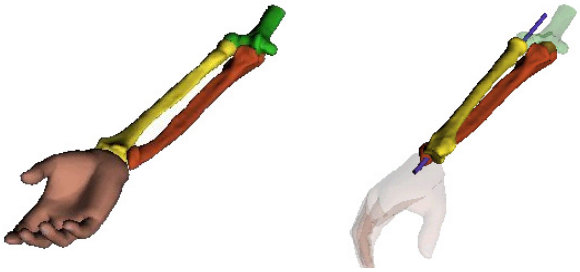
on consider them to be motion in the transversal plane (Figure 3.7(c)) rather than in the sagittal plane. The reason for this is that in computer graphics and vision applications, the resting pose for the arm is generally defined as being abducted by $\frac{\pi}{2}$ radians, palm facing downwards, in order to avoid rotation singularities that generally arise at multiples of 2π. For a discussion of 3D rotation representations and their singularities, see the Appendix. With the arm-outstretched initial pose, these singularities are avoided. However, the drawback is that the medical definition of flexion/extension is no longer applicable.

3.2.2 Upper limb anatomy

3.2.2.1 Shoulder complex

The shoulder compound is constituted of three joints, all of which have 3 DOFs, as shown in Figure 3.9, and can be modelled as ball-and-socket joints:

- the sterno-clavicular joint that moves the clavicular bone;
- the acromio-clavicular joint that moves the scapula;



(a) (b)

Figure 3.10: The lower arm rotation: (a) resting position of the lower arm; (b) rotation of the lower arm, with rotation of the radius around the ulna [IST1999].

- the gleno-humeral joint that allows motion of the humerus (i.e. the upper arm).

All joints have complete mobility, as all have 3 DOFs. We would also like to point out the presence of what is commonly referred to as the scapulo-thoracic joint, despite the fact that it is not really a joint in the proper sense of the term [Maurel1998]. This joint allows the gliding of the scapula on the thorax.

The motion of these various joints is understood to be inter-dependent and is in general referred to as the "shoulder rhythm", which defines the amount of rotation of each joint with respect to upper arm elevation. These joint inter-relationships have been the subject of various research aiming at quantifying them, as in [DeGroot and Brand2001] and [Moeslund *et al.*2003].

3.2.2.2 Elbow joint

2 DOFs are defined at the elbow level, these being referred to as forearm pronation and supination in the biomechanical domain. The joints involved are both 1 DOF hinge joints:

- the ulno-humeral joint that moves ulna and radius together (i.e. the lower arm) in a flexion motion;
- the ulno-radial joint that rotates the radius around the ulna (Figure 3.10) thus performing axial rotation of the lower arm;

the motion components of these joints being independent from each other. This being the case, we can represent rotation at the elbow level as a single 2 DOF joint having one swing (flexion/extension) component, as well as twist (axial rotation).

3.2.3 Simplistic shoulder model

To the best of our knowledge, all body models used in video-based motion capture rely on an over-simplified shoulder joint model, on the basis of the general assumption that more detail is unnecessary, that it would be computationally expensive, but most of all, that it is sufficient. The elbow, of course, is not the subject of any such issues, as its motion components do not present any difficulty in terms of representation.

The common trend for simplifying the structure of the shoulder region is more or less similar to the ones shown in Figures 3.3, where we can see that one single joint has been considered as representative of the motion of the entire shoulder compound.

In such a simplified model, the shoulder joint is modelled as a 3 DOF ball-and-socket joint (Figure 3.2(e)). This basically boils down to ignoring the three other joints of the shoulder complex and considering all motion to

be concentrated in the gleno-humeral joint. This representation is widely used in the computer vision community, and sometimes even in computer graphics, when realism is not an issue.

When the 3D data available to us for retrieving the posture of our model is sparse or generally insufficient in order to recover detail at the shoulder level, we will also use this ball-and-socket simplification to represent the shoulder joint.

3.2.4 Anatomically-correct shoulder model

At the other extreme, in the event where we will be matching a model to stereo data representing the 3D contour of the captured person, we will have at our disposal an ample amount of 3D data, therefore allowing us a more detailed body model. The two-fold purpose of this is to increase the realism of the retrieved animation and to minimise the ambiguity of the posture recovery process. In this perspective, we will underline the limitations of the simple ball-and-socket model, illustrating its short-comings with some specific examples .

In practice, only a sub-set of the shoulder joints are observable. A couple of years back, the Delft Shoulder Group proposed a measurement protocol of the shoulder joints designed for an electro-magnetic tracking system [Meskers *et al.*1998]. Later, the same group published a standardised motion recording protocol for the shoulder [VanDerHelm1997] and explained how the protocol could be extended to video-based recording. However, it is clearly stated that "due to the large bone-to-skin displacements, no clavicular or scapular motions can be recorded using external markers", fact that is confirmed in the context of further biomechanical research work on the shoulder area, such as [Bao and Willems1999]. The two extremities of the clavicular bone are easily located and one would think that therefore, they can be readily tracked, but as we will experience first-hand, the large bone-to-skin displacements inevitably do appear. In any case, neither axial rotation of the clavicula nor any motion of the scapula can be determined from video. If need be, the position and orientation of the clavicula and scapula can be inferred using linear regression, on the basis of (1) the visible bony landmarks and (2) the initial scapula position, which needs to be measured manually at the beginning of the recording [Veldpaus *et al.*1988]. What information we derive from the previously-listed studies is detailed in the following sub-sections.

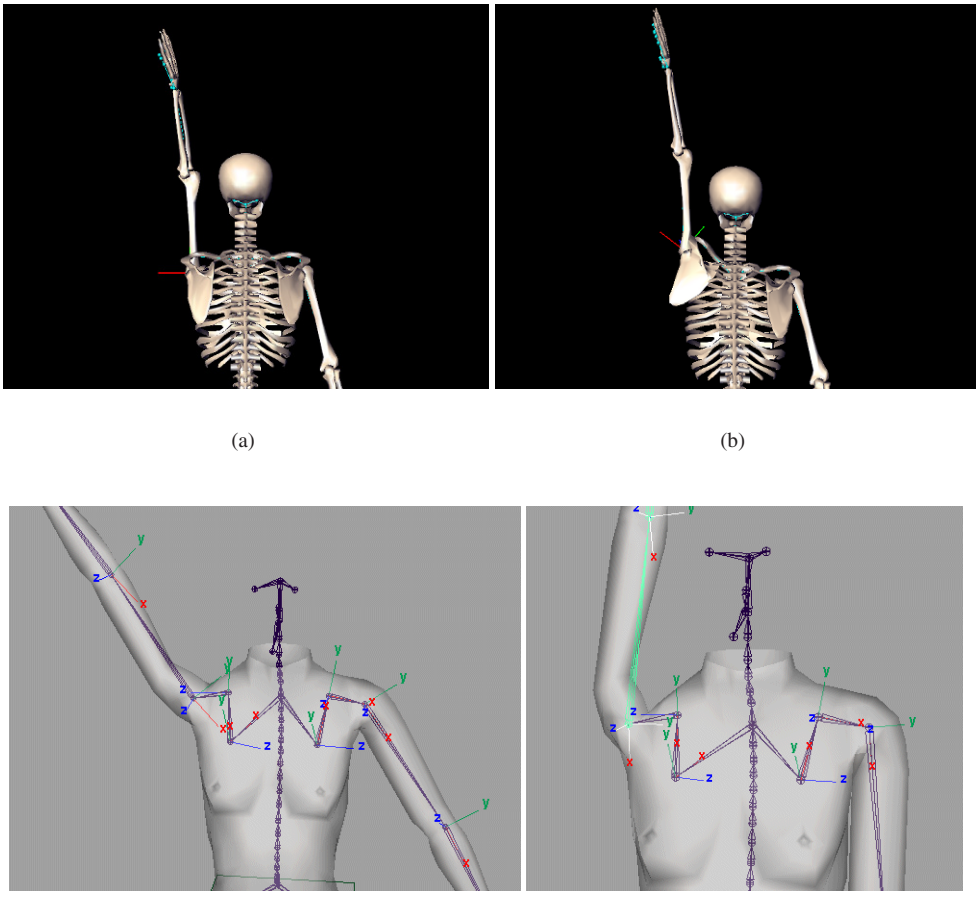
3.2.4.1 Clavicular motion

Axial rotation of the clavicular bone is neither visible on the outside level nor does this motion component of the sterno-clavicular joint directly imply any deformations at the surface level, i.e. of the soft tissues and skin. Paradoxically, axial rotation of the clavicula is its largest rotation, ranging from 0 to $60\mathrm{deg}$ [VanDerHelm and Pronk1995]. The directly connected acromio-clavicular joint carries out axial rotation in harmony with the sterno-clavicular joint, thus rotating the entire shoulder complex, including the gleno-humeral joint. We will therefore freely state that the axial rotation measured at the gleno-humeral joint is identical to the one at the two other joints, as these move pretty much as a whole in terms of twisting.

If axial rotation does not affect skin deformations, there is nevertheless a need for the possibility of angular rotation at the sterno-clavicular level. On the one hand, the absence of clavicular motion results in unrealistic body shapes at this level (Figure 3.11), and on the other hand, does not allow the representation of joint inter-dependency, namely in terms of upper arm elevation (Figure 3.12).

3.2.4.2 Scapular motion

It was stated in the introductory paragraph of this sub-section that scapular motion was not visible from the exterior, meaning that it is not truly necessary to model the two joints (scapulo-thoracic and acromio-clavicular) if their motion has hardly any repercussion at the surface level. This would seem as a justified assumption unless the surface data used for motion recovery is extremely precise and reflects the punctual saliency of the scapula in the cases of extreme upper arm extension and/or particularly skinny subjects. The scapula being a bone set into motion by two joints, modelling it would require a real effort and is hardly justified except in terms of computer character animation using an anatomically-realistic body model.



(a) (b)

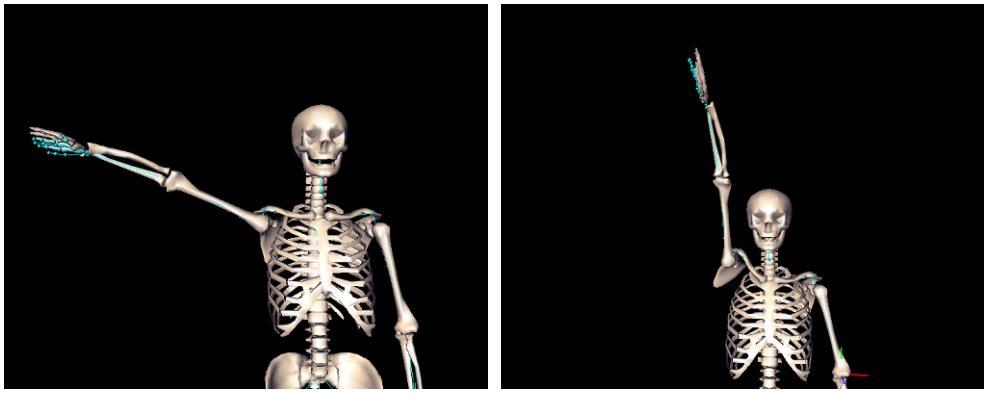
(c) (d)

Figure 3.11: Model with or without clavicular mobility: (a) The mobility of the shoulder compound is limited to the gleno-humeral joint, illustrated in the case of a simple motion of raising the arm, which does not reflect reality in the least; (b) if clavicular mobility is added, the skeleton has a realistic look. (c) and (d) When the shape-defining layers are deformed on the basis of the underlying skeleton, the resulting deformation would be grossly unrealistic in the case of a ball-and-socket joint, and would certainly not correspond to surface data.

As said before, the scapula has altogether 5 DOFs, two of which allow it to slide to and fro on the thorax. In practical terms, rather than defining two joints of respectively 3 and 2 DOFs to model scapular motion, [Maurel and Thalmann2000] have chosen to implement a 3 DOF joint for defining its 3D rotation. On top of this, the scapulo-thoracic constraint ensures that the scapula stays at all moments in contact with the thorax, thus preventing it from sticking out at awkward angles (Figure 3.13).

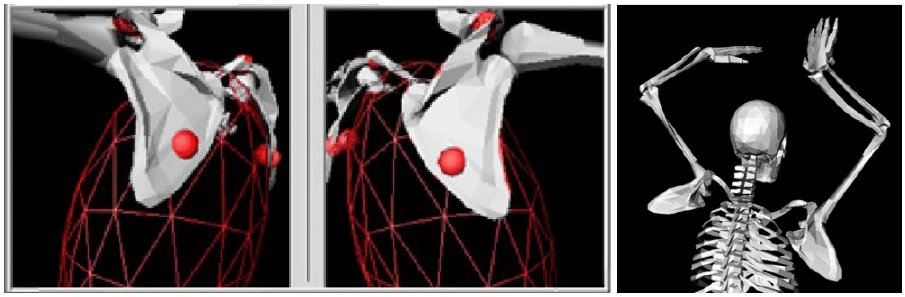
3.2.5 Our model

For tracking applications, joints whose motions is not visible on the outside represent unnecessary complexity and detail, the fact that they are extremely impractical to measure being a direct consequence. We have therefore decided to retain only the sterno-clavicular and gleno-humeral joints for representing the shoulder complex, i.e. the joints whose motion can be readily seen and measured.



(a) (b)

Figure 3.12: Coupled-joint model: (a) Effective maximal arm elevation without involving motion of the clavicula. As one can see, without contribution of the clavicular bone, gleno-humeral joint elevation is relatively limited; (b) The sterno-clavicular joint contributes to arm elevation and provides the additional mobility.



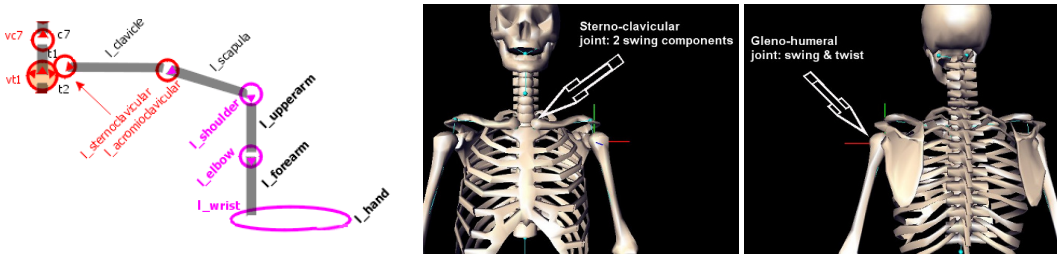
(a) (b)

Figure 3.13: The scapulo-thoracic constraint: (a) The constraint is enforced as a point on the scapula being in contact with the thorax at all times; (b) Case of motion without scapulo-thoracic constraint, and the resulting phenomenon of the "flying scapulas" [Maurel1998].

In reality, the sterno-clavicular and gleno-humeral joints have 3 degrees of freedom each, but as we mentioned before, the twisting component of the clavicula will be discarded as it is directly translated to gleno-humeral joint twist. We can therefore consider the sterno-clavicular joint as 2D ellipsoidal (two swing components), the gleno-humeral as 3D ball-and-socket joint (two swing and one twist components) and the elbow either as a 1D hinge joint (one swing component) or a 2D pivot joint (one swing and one twist component).

The choice of the elbow joint type is arbitrary: either the elbow is modelled as a 1 DOF joint and the wrist with 3 DOFs, or both joints have 2 DOFs. It is just a question of which joint should inherit the twist component of the lower arm, which in reality corresponds to the rotation of the radius around the ulna, and is therefore a bone rotation rather than a joint rotation.

In our case, we have chosen to model the elbow joint in 2D based on the simple observation that if the lower arm is blocked in one way or another, no rotation is possible at the wrist level, and therefore, it seemed logical to bestow the twist component onto the elbow joint. However, the 1D elbow/3D wrist representation is just as common [Lee2000]. We show in Figure 3.14(a) our final upper limb model having the degrees of freedom



(a) (b) (c)

Figure 3.14: The joints at the shoulder and upper arm level: (a) Articulated structure corresponding to the human upper limb in the H-Anim standard; (b) front view showing the acromio-clavicular and elbow joints; (b) back view showing the gleno-humeral joint.

illustrated in Figure 3.14(b) and (c).

We have thus defined a body model usable for fitting to dynamic 3D data, its joint structure and degrees of rotational freedom determining the state-space of possible postures that the model can adopt. It is of course perfectly clear that anatomically-speaking, the set of postures that a human being can physically take on is a sub-set of the state-space solely defined by its rotational parameters. The state-space of postures is nevertheless naturally reduced by increasing a model's realism and detail, which is what we have endeavoured to do by defining a more anatomically correct upper limb representation. Is is however not necessary, in terms of motion capture, to further increase its complexity as the efficiency of posture recovery would not be enhanced through it. On the contrary, it might even be hampered by the additional parameters whose value would need to be determined. A more meaningful way of reducing the state-space of posture solutions is to add various constraints, either extrinsic or intrinsic to our body model, the usefulness of this being justified in the following chapters.

Chapter 4

Imposing extrinsic motion capture constraints

The main problem of pose recovery in the context of motion capture is that several configurations can produce the same observed data, therefore forcing us to come up with one means or another of determining which of the configurations is the most plausible.

What defines the notion of "plausible" is a set of conditions that must be satisfied. In the state-of-the-art on motion capture presented in the second chapter, the constraints used for pruning the solution space mainly fall into two categories, namely intrinsic or extrinsic to the body model. The latter constraint type is applicable in video-based motion capture from the moment that multiple cameras are present, the disambiguating of postures being its purpose.

Extrinsic constraints are by definition provided by elements outside the body model, these including:

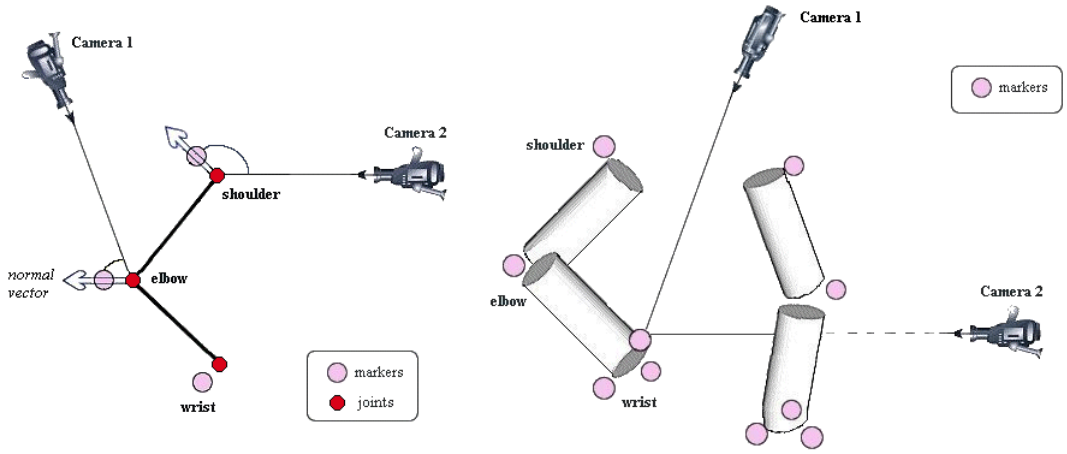
- smoothness assumptions, such as thresholds on frame-to-frame motion velocity and acceleration, these being directly linked to the sampling frequency of the cameras;
- consistency with learned motion patterns, thus confining the search process to a certain type of motion;
- visibility based on knowledge of camera position and orientation.

Constraints such as learned motion patterns will not be considered in the current work, as we wish to be able to deal with arbitrary motion.

As we have mentioned before, stereo data extracted from multiple views can either correspond to 3D reflective marker positions for infrared cameras, or to 3D surface data in the case of CCD cameras. We will illustrate the usefulness of extrinsic constraints in the context of optical motion capture, as such an application will most benefit from this type of constraints. Indeed, accurate stereo reconstruction and data segmentation are paramount, due to the fact that the 3D data is sparse and that an erroneous reconstruction or identification will inevitably lead to incorrect posture recovery. Extrinsic constraints will in this case greatly enhance not only the process of estimating motion parameters, but also the prior steps of stereo triangulation and data segmentation. We will furthermore show how these constraints not only increase the robustness of each step of the process, but also how coherence constraints between these various steps can prune the search-space of solutions causing the amount of posture recovery error to drop even more.

4.1 Extrinsic constraint types

In this section, we will be defining the various categories of extrinsic constraints that can be applied to video motion capture, these constraints being integrated into our motion capture process, as will be detailed in the



(a) (b)

Figure 4.1: (a) *Visibility constraint:* the point on the elbow is clearly visible to camera C_1, as the normal at the point with respect to the underlying skeleton produces an acute angle with the ray connecting the optical centre of the camera to the elbow joint. The point at the shoulder joint, on the other hand, is not visible from camera C_2, as the angle formed is obtuse; (b) *Occlusion constraint:* the point on the left wrist is visible from camera C_1, as the line connecting the optical centre of the camera with the point on the wrist does not intersect any of the body solids. The point is however occluded by the right lower arm body part, obstructing it from the view of camera C_2.

subsequent sections.

4.1.1 Stereo vision

The additional information provided by camera set-up knowledge is precious when it comes to disambiguating 3D model postures on the basis of multiple camera views. Given that a clear correspondence exists between pixels in a camera view and 3D points in the scene, camera parameters have become an asset in terms of motion capture, and are systematically integrated into the process, when available. We refer the reader unfamiliar with the notion of stereo vision to the Appendix.

4.1.2 Visibility constraint

The first constraint that can be derived from the principle of stereo vision is the visibility constraint, which aims at determining whether a given 3D point in the scene can be seen from a given camera. Such a constraint is enforced, for example, in order to ensure that we are not matching the wrong side of a body model to the data. An example of the visibility constraint is shown in Figure 4.1(a).

4.1.3 Occlusion constraint

Using the optical centres of the cameras around the scene, one can also ensure that a given 3D point is not obscured from view by some body part of the model. Such a constraint of course requires the body model to have a shape approximation. The constraint consists simply in tracking a line from the 3D point to the position of the camera, and to test for intersection with all body parts (see Figure 4.1(b)). How easily this can be done depends on the chosen shape representation.

4.1.4 Motion smoothing

For a given camera sampling rate, one can set a threshold on the maximum possible displacement of a human body part. A few methods have integrated such a smoothing constraint, namely [Plänkers and Fua2001, DiFranco *et al.*2001, Dockstader and Tekalp2002].

4.2 Applying extrinsic constraints

For applying and evaluating the impact of extrinsic constraints on the fitting of a body model to dynamic 3D data, we have developed a method that incorporates such constraints at every level of the motion capture process, which we recall are:

1. 3D reconstruction,
2. data segmentation,
3. posture recovery.

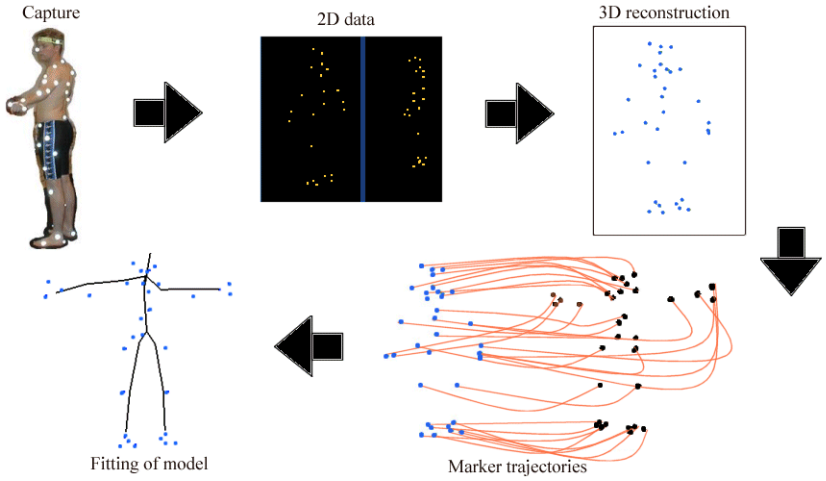
The proposed method is illustrated - but not limited to - the case of optical motion capture, and aims, through the combined use of a body model and extrinsic constraints, to solve the inherent difficulties that the set-up presents, which were laid out in Section 1.3.

4.2.1 Introductory comments

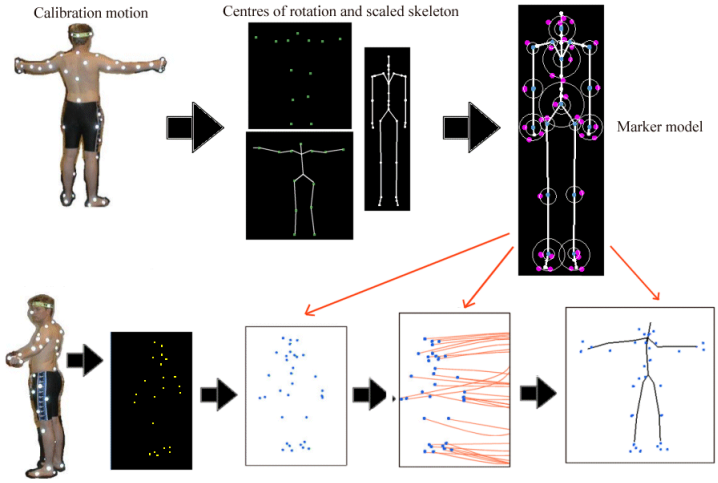
In the case of optical motion capture, the items to be recognised and tracked being limited to markers, if these can be reconstructed and identified accurately, then posture recovery is relatively simple. However, even with a highly professional system, there are many instances where crucial markers are occluded or when the algorithm confuses the trajectory of one marker with that of another. Such labelling errors are at the source of the post-processing efforts involved on behalf of the motion capture operator. To reduce this overhead, we propose the use of an approximate anatomical human model, such as the one described in Chapter 3, in order to accurately predict the 3D location and visibility of markers. In this manner, the robustness of marker tracking will significantly increase and the need for human intervention during the 3D reconstruction process drastically reduced or even eliminated.

Several major difficulties arise in the course of the process. First of all, the reconstruction of the markers in 3D must be accurate: this does not present great difficulties in the case where markers are visible from at least two cameras, but in the presence of occlusions, markers may be lost. The markers then need to be tracked from one sequence frame to the next. The identification of these markers implies real difficulties, and there are many instances where markers are occluded or when the algorithm confuses the trajectory of one marker with that of another. In the case where markers become occluded over a certain time span, it is necessary to one way or another infer the intermediate marker positions as otherwise body posture recovery is impossible. As to marker trajectory confusions, these typically occur because these markers are attached to skin or tight clothes that have their own relative motion with respect to the underlying bone structure. Indeed, if the marker identification process expects constant inter-marker distances or marker-to-joint distances, problems are guaranteed to arise. The final obstacle, once the markers have been identified, is to correctly associate the body model with the set of 3D markers and infer the corresponding posture.

The short-comings of the previously mentioned process (see Figure 4.2(a)) limit its applicability in a real-time context and drive up post-processing costs for non real-time applications. This is the issue that our proposed approach - depicted by Figure 4.2(b) - addresses. In most commercially available packages and research work, the estimation of the 3D marker positions and the fit of the 3D model are decoupled, whereas we first compute a body-and-marker model using a standardised set of motions. We then use it to resolve the ambiguities during the 3D reconstruction process.



(a)



(b)

Figure 4.2: Approaches to motion capture: (a) usual approach; (b) our approach.

In contrast to the habitual process, we do not treat 3D marker reconstruction independently from motion recovery either. Instead, we combine these two processes and use prediction techniques to resolve ambiguities. For example, we can predict whether or not a marker is expected to be occluded by the body in one or more images and take this knowledge into account for reconstruction purposes. When a marker cannot be reconstructed with certainty from its image projections, we use the expected position of the skeleton to identify the marker and disambiguate its 3D location. This is helpful when it is only seen by a small number of cameras. In our approach, skeleton posture recovery is a by-product of the reconstruction process.

In the following sub-sections, we present the theory behind 3D marker reconstruction, tracking and identification, then moving on to describing the actual processing flow applied to measured data, and then present the outcomes. We also provide some analysis of the system's performance, demonstrating its robustness using some

complex motions that feature both large accelerations and severe occlusions.

The two major steps leading from a captured motion to a reconstructed one are:

- marker reconstruction from 2D marker sets to 3D positions;
- marker tracking from one frame to the next, in 2D and/or 3D.

However, despite the fact that 2D and 3D tracking ensure the identification of a large number of markers from one frame to another, ambiguities, sudden acceleration or occlusions will often cause erroneous reconstructions or breaks in the tracking links. For this reason, it has proven necessary to increase our procedure's robustness by using the skeleton to drive the reconstruction and tracking process by introducing a third step. The latter consists in accurately identifying each 3D marker and carrying out a complete marker inventory in each frame.

The set-up used for the process consists of a certain number of infrared cameras that surrounds the scene (a typical number would be 8 or 9), their intrinsic parameters being known, the output being nothing more than the 2D positions of each marker effectively seen in each camera view.

4.2.2 Body-and-marker model

This body model we are using here is the simpler version of the two presented in the previous chapter, where detailed hands and feet have been excluded, as the number of parameters of the model must remain lower than the actual number of 3D data. This model has a marker model associated to it, where the markers are attached to specific joints and are constrained to remain on a sphere centred in that joint. Incorporating information regarding the relative positioning of data with respect to the model has been used previously. This was the case in [Theobalt *et al.*2002], where point samples representing the body part surface were attached to the body model, and in [Wilhelms *et al.*2000], where the parameters of the snakes representing 2D body part contours were also stored within the model itself.

In our case, representing the relative motion of a marker around a joint by a sphere obviously is a simplistic representation that does not reflect the effective motion. If more precision were necessary, this model could readily be refined so as to describe more exactly the motion of a marker with respect to its underlying joint. This marker model is dynamically computed on the basis of the current motion capture session, as the marker configuration varies for each actor. The skeleton layer of the body model and its associated marker model are shown in Figure 4.3.

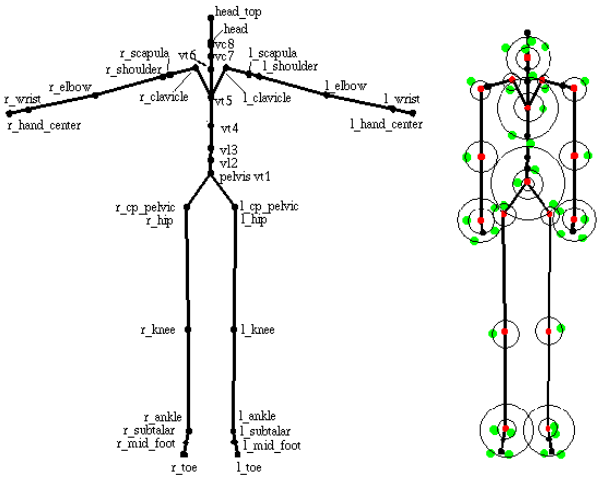
4.2.3 3D marker reconstruction

Stereo triangulation is used to reconstruct 3D marker positions. Given the projection P_1 of a point in an image, the corresponding projection in another image must lie on the epipolar line. The correspondence between the two images is established by the fundamental matrix, computed on the basis of the calibration data of the cameras.

We perform the test of the epipolar constraint on a set of two views, thus performing pair-wise reconstruction. For each non-ambiguous stereo match, that is when there is only one possible candidate, we compute the 3D co-ordinates on the basis of the 2D co-ordinates. The 2D markers whose co-ordinates were used for reconstruction are assigned an associated 3D index corresponding to the created 3D marker. This principle is shown in Figure 4.4.

These 3D co-ordinates are then re-projected onto the remaining camera views, to determine the corresponding 2D markers. These 2D markers found on re-projection are assigned a secondary 3D index corresponding to the reconstructed 3D marker. In this manner, for each 3D marker, we can determine the complete set of corresponding 2D markers in the camera views.

We assume that a 3D marker is correctly reconstructed if it re-projects into at least one other camera view (thus making a total of three camera views). We will say that these markers are reconstructed by trinocular stereo, i.e. using at least three cameras. This is in contrast to markers reconstructed using only two camera views, and for which projections in the other views could not be found.



(a) (b)

Figure 4.3: (a) Static human skeleton model; (b) marker model associated to the skeleton.

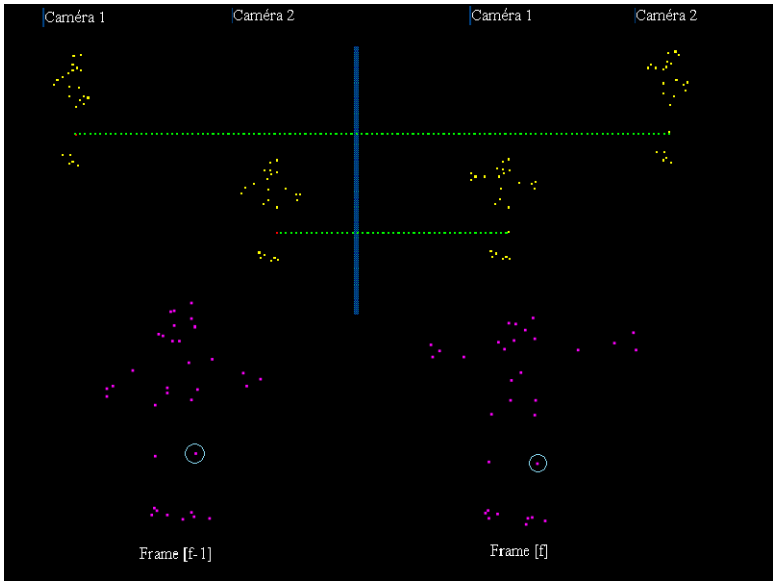
4.2.3.1 Binocular reconstruction

Once we have reconstructed these trinocular 3D markers in the first frame, we need to compare the number of reconstructed markers with the number of markers known to be carried by the subject. As all remaining processing is automatic, it is absolutely essential that all markers be identified in the first frame. Any marker not present in the first frame is lost for the entire sequence. Therefore, if the number of reconstructed markers is insufficient, a second stereo matching is performed, this time also taking into account markers seen in only two views.

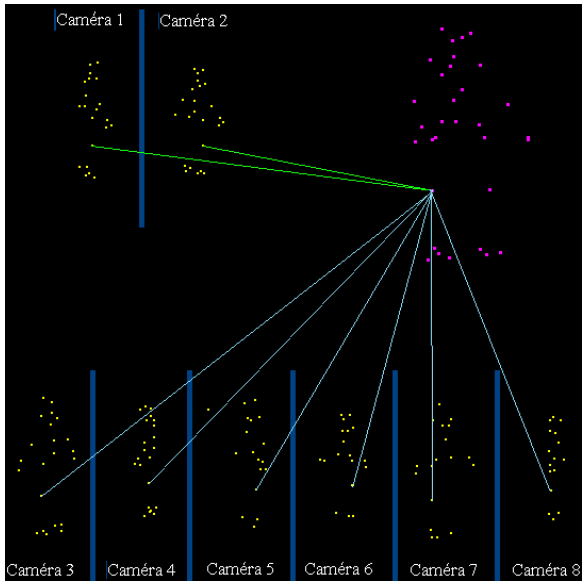
4.2.3.2 Testing the accuracy of the reconstruction using constraints

In order to improve the results of stereo matching, the first essential step for accurate pose recovery, we will apply camera-related constraints to verify the validity of the reconstruction. More specifically, we use the skeleton for applying a visibility and occlusion test to each pair of 2D markers used to construct a 3D marker.

Visibility constraint A marker is expected to be visible in a given view if it is seen more or less face on as opposed to edge on, that is if the surface normal at marker location and the line of sight form an acute angle. Suppose that we have reconstructed a certain 3D marker using the 2D pair (marker i_1, view j_1) and (marker i_2, view j_2). We check that these two markers i_1 and i_2 are indeed visible in views j_1 and j_2 respectively. Given the fact that the motion capture sampling rate is in general at least 100 Hz, we can safely assume that displacement from one frame to another is minimal. Hence, we use the posture of the skeleton posture in the previous frame and calculate the normal at the 3D marker location with respect to its underlying body part segment. We draw the line joining the 3D marker co-ordinates to the position in space of the camera and if the angle between the normal and the line is acute, then the marker is visible. If this test shows that we have used the wrong 2D co-ordinates for reconstruction, we must select other candidate 2D co-ordinates. As mentioned before, each 3D marker is associated to two sets of 2D co-ordinates determined by stereo correspondence, which we then use for reconstructing the 3D marker. To this 3D marker, we then also associate the 2D co-ordinates from the remaining camera views onto which the 3D co-ordinates of the marker projected correctly. Given that the visibility test has detected an erroneous 3D reconstruction, we choose one of the 2D co-ordinates computed via 3D to 2D



(a)



(b)

Figure 4.4: (a) Reconstruction of a 3D marker from two camera views, and (b) re-projection onto the remaining views to find the associated 2D markers.

projection, and calculate new 3D co-ordinates, as shown in Figure 4.5. We then apply the constraint again, and if this fails, we repeat the entire procedure until we find a correct reconstruction set.

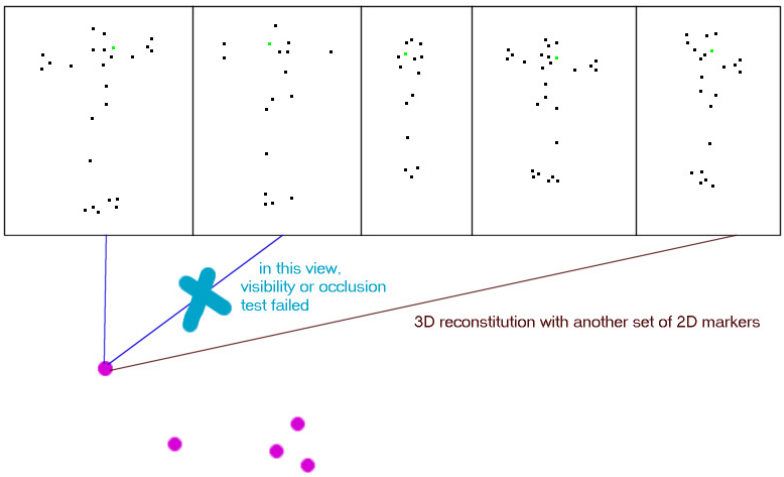


Figure 4.5: Safe 3D marker reconstruction from validated 2D markers.

Occlusion constraint Once a 3D marker has satisfied the visibility constraint, it also needs to comply with the occlusion constraint that ensures that the 3D marker is not occluded from some camera views by some body part. To this end, we approximate body parts by solids, using simple geometrical primitives, namely cylinders for limbs and a sphere for the head. For each 3D marker, a line is traced from the marker to the position of the camera, and tested for intersection with all body part solids. In case of intersection with a solid, there is a non-zero probability that the marker is occluded from this camera view. In order to not take the chance of creating an erroneous association, we prefer to choose other 2D co-ordinates and repeat the process until all constraints are satisfied.

4.2.4 Tracking

Once as many markers as possible have been reconstructed, we can proceed with tracking of these markers from one frame into the next, thus resulting in marker trajectories over the entire sequence. 2D tracking is carried out at the same time as 3D tracking because 2D sequences are bound to provide more continuity than reconstructed 3D sequences. We therefore use 2D tracking in order to accelerate 3D reconstruction. For each reliably reconstructed marker in frame [f], we consider the two sets of 2D co- ordinates that were used to compute its 3D co-ordinates. After 2D tracking, these two sets of 2D co-ordinates will most likely have links to two sets of 2D co-ordinates in [$f+1$], the next frame. If so, we can then use them in [$f+1$] to construct the corresponding 3D marker. To determine the related 2D positions in the other camera views, we re-project the 3D co-ordinates, as in the stereo matching process described above.

3D tracking propagates the information attached to each marker in the first frame throughout the entire sequence, so that as many markers as possible are identified in all frames. A broken link in the tracked trajectory of a marker would imply the loss of its identity and a subsequent user intervention. In the next sub-section, we will see how we use the skeleton to overcome that problem in an automated fashion. To compute the trajectory of a marker from frame [f] into frame [$f+1$], both in 2D and 3D, we look at the displacement of the marker over a four-frame sliding window (Figure 4.6) as per the particle tracking in 3D flows algorithm of [Malik *et al.*1993].

The basic assumption is that displacement is minimal from one frame to the next, and the idea is to predict and confirm the position of a marker in the next frame. The displacement of a marker from [$f-1$] into [$f+1$] predicts the position in [$f+1$]. The actual position in [$f+1$] and the projection of the movement into [$f+2$] should confirm the previously made hypothesis by eliminating ambiguities. In other words, we follow the displacement of markers over a sliding window of four frames.

Three cases can arise for each marker in [f] that we attempt to track into [$f+1$]:

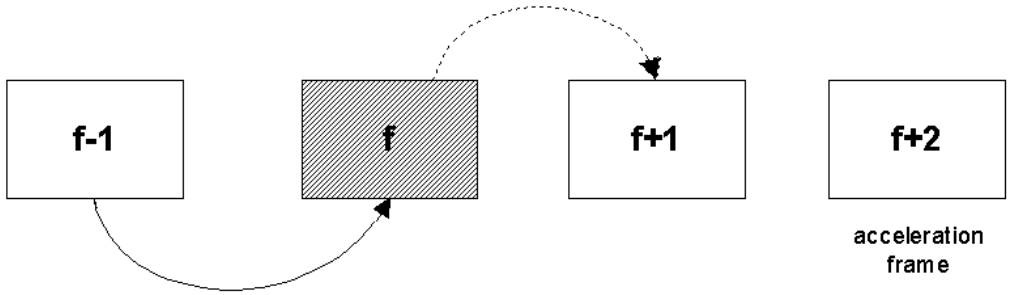


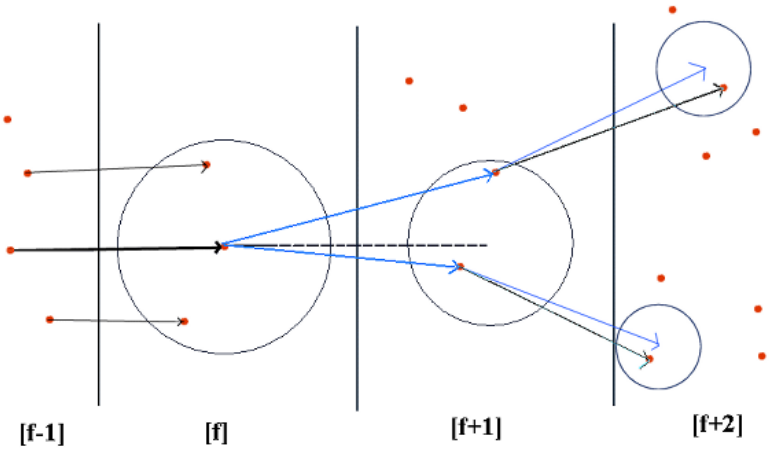
Figure 4.6: Using a sliding windows of four frames to perform tracking, where [*f-1*] is the previous frame, [f] the current frame, [*f+1*] the frame in which we predict the next position and [*f+2*] the acceleration frame that will eliminate ambiguities in the case of several prediction candidates.

- **The marker has a tracking link from [*f-1*] into [f]:** in this case, we calculate the displacement from [*f-1*] to [f] and apply it to the current position of the marker in [f], maintaining the direction of the movement. The new position x_i^{f+1} of a marker x_i in frame $[f+1]$ is therefore defined as: $x_i^{f+1} = x_i^f + v_i^f.\Delta t$, where v_i is the velocity of the marker computed over the time span Δt. Once we have a predicted position, we define a search neighbourhood in [*f+1*] centred in the predicted position, this being the region in which we expect to effectively locate the marker. The radius of such a neighbourhood is function of the currently observed acceleration, thus expanding when the movement accelerates. If within this neighbourhood we find more than one candidate, we prolong the movement into [*f+2*] in exactly the same fashion, for each candidate in [*f+1*]. The trajectory from [*f-1*] to [*f+2*] that has the smoothest acceleration will determine the effective track from [f] into [*f+1*].

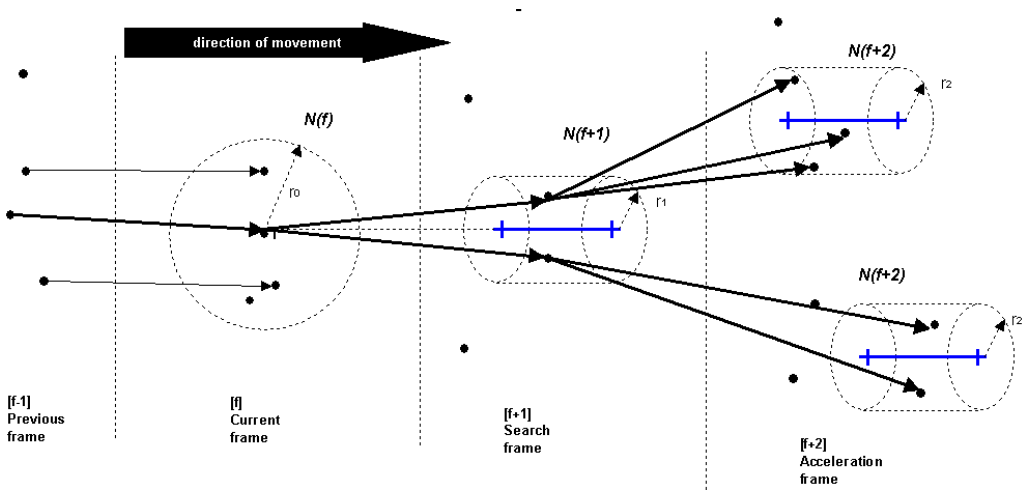
- **The marker has no tracking link:** we define a correlation neighbourhood in [f], centred in the current position of the marker in [f]. This neighbourhood includes markers whose movement is correlated to the marker in focus. In the original particle tracking algorithm, this correlation neighbourhood is defined as being circular, but in the case of human body tracking, the correlation neighbourhood would typically be defined as the set of markers on a same limb, as these are bound to have the same speed of motion. We use the links from [*f-1*] into [f] of those markers to determine the displacement to apply to the marker. We then proceed as above.

- **There is neither of the above:** we can only define an arbitrary displacement (based for example on the average displacement that we expect for markers from one frame to another), and proceed as above. However, in order not to unnecessarily introduce errors, we prefer to perform tracking in two passes, identifying and tracking markers with a link or a correlation neighbourhood, and then in a second pass deal with the remaining markers. As tracking identifies markers in the next frame, we will know to which markers the untracked markers correspond to, as we will perform an inventory. This should then easily allow us to define a correlation neighbourhood by using the markers on the same limb.

In all cases, the predicted position is not considered to be a fixed point, but a segment whose extremities are the predicted position assuming zero displacement, as there is not necessarily motion from one frame to the next, and the predicted position calculated by one of the three above-mentioned schemes. We assume that the effective position will be on this segment. If one uses only the predicted position on the basis of acceleration, errors will be introduced in the case where the subject suddenly remains immobile from one frame to another. Indeed, if we apply the acceleration observed in the ***n*** previous frames to a marker in the current frame, this will obviously yield a marker displaced with respect to its previous position. However, if the human subject is no longer moving but is immobile, the predicted position will be inaccurate and will produce errors in the reconstruction.



(a)



(b)

Figure 4.7: (a) Illustration of the four-frame tracking principle, where [*f*] us the current frame that we are tracking into frame [*f+1*], [*f-1*] is the previous frame enabling us to construct a correlation neighbourhood, and [*f+2*] being the acceleration frame that allows the sorting of the candidate tracks; (b) enhanced four-frame tracking using a segment of predicted positions to also take into consideration an eventual immobility of the subject.

In the example of Figure 4.7, the marker in focus in [*f-1*] has a link in [*f-1*], on the basis of which we prolong the movement into [*f+1*]. In the circular search neighbourhood formed, there are two possible candidates, and we are therefore faced with an ambiguity. To solve this, we prolong the movement into frame [*f+2*], thus obtaining a trajectory for each candidate. Of the two trajectories, we favour the one with the least change in acceleration over the four frames, i.e. where

$$\Delta a = a_{[f]\rightarrow[f+2]} - a_{[f-1]\rightarrow[f+1]}$$

is smallest, this then determining the marker in [*f+1*] that is indeed the correct link to the one in [*f*]. Once the link is established, we verify that it is consistent with the simultaneous 2D tracking carried out in the camera views. Any inconsistent link will not be established and the corresponding marker will be discarded momentarily, in order to be dealt with in the second tracking pass.

Figure 4.7(b) shows a similar example, but the predicted position in the search frame is now located within a flattened cylinder and no longer within a circle, based on the assumption that motion can also cease from one frame to the next.

4.2.5 Marker identification and inventory

The identification of all markers is necessary in order to be able to associate the skeleton model to the cloud of markers. However, tracking alone is mostly not sufficient for this purpose, due to tracking limitations, noise in the measured data, etc. Combining the model and marker tracking should allow us to complete the job, as well as to perform a marker inventory -as each marker needs to be present in each frame - and to infer the position of eventual missing markers.

Say we have just performed 3D reconstruction using the 2D data of frame [*f*], and have thus obtained a set of markers. We then proceed with the verifications detailed in the following sub-sections, in the order in which they appear.

4.2.5.1 Binocular reconstruction

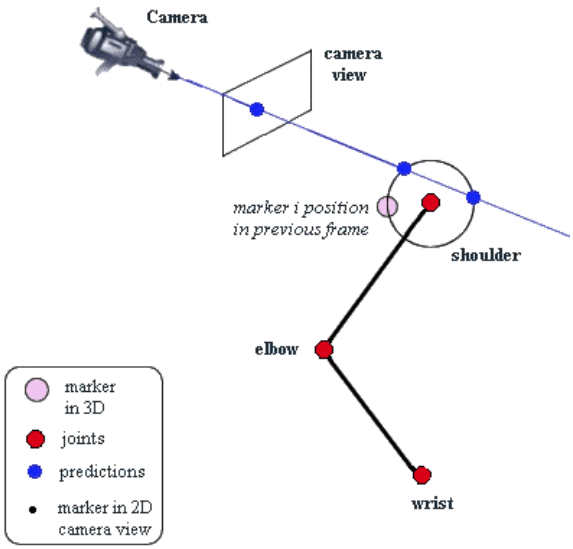
If the number of markers reconstructed using trinocular stereo is smaller than the actual number of markers worn by the subject, we perform binocular reconstruction and add the newly calculated co-ordinates to the already existing list of markers.

4.2.5.2 Identification by tracking

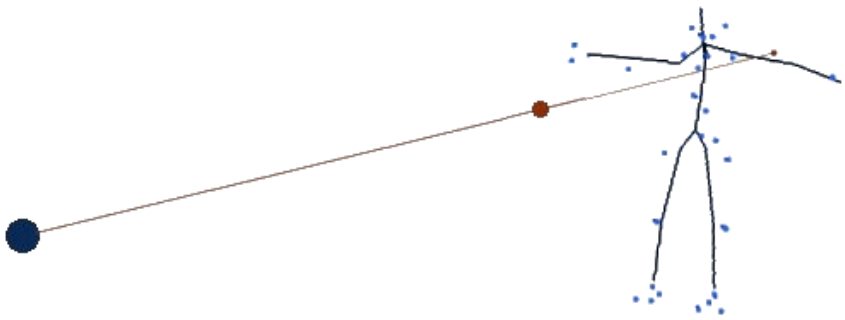
We perform 3D tracking from [*f-1*] into [*f*], thus identifying a certain number of markers in [*f*], i.e. attaching them to their legitimate joint. 3D tracking is performed in two passes, the first one being regular 3D tracking with the predefined search neighbourhood, and the second pass attempts to track the still unlinked markers but allowing expansion of the search neighbourhood by a certain factor. As the neighbourhood is statically set by the user upon initialisation, and its value is modified over the sequence, based on the computation of current displacement speed.

4.2.5.3 Monocular reconstruction

If the number of reconstructed markers is still smaller than the actual number of markers worn by the subject, we resort to monocular reconstruction. For this, we browse through the set of 2D markers that are not associated to any reconstructed 3D marker, and this for all cameras. For each such 2D position, we calculate the optical centre of its associated camera, and calculate the equation of the line emanating from the camera and passing through the 2D position, thus giving us a line in 3D global space on which the corresponding 3D marker should be located. For each marker that still remains to be localised in the current frame, we retrieve its position in the previous frame, as well as the position, in the previous frame, of its underlying joint. We then retrieve its marker-to-joint distance and create a sphere centred in the joint of radius marker-to-joint distance. If the line emanating from the optical centre intersects (at one or two points) with this sphere, then we are able to re-create the missing marker. In the case of one intersection, there is no ambiguity. In the case of two intersections, we will retain the co-ordinates that most closely match the position of the marker in the previous frame. In the case of monocular reconstruction, the reconstructed marker is already identified, we have therefore separated it from the trinocular and binocular reconstructions, as no tracking is necessary (Figure 4.8).



(a)

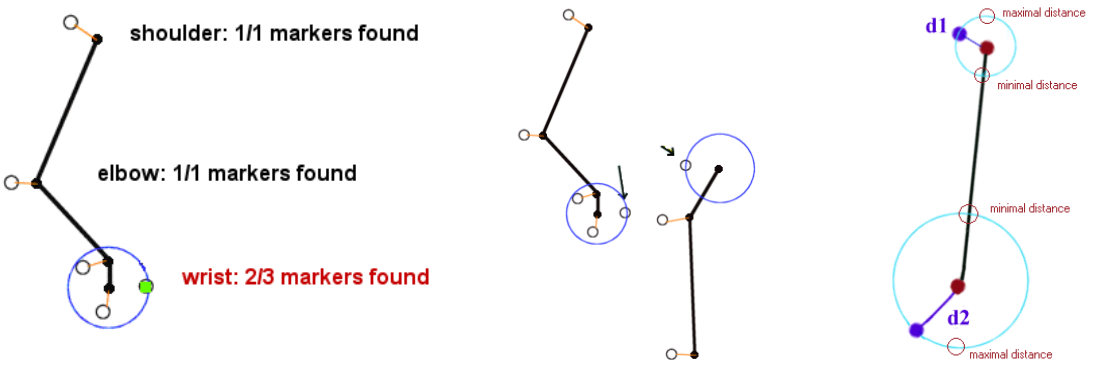


(b)

Figure 4.8: (a) One-camera marker reconstruction principle: the ray cast from the optical centre of the camera through the image point of marker ***i*** intersects (in world space) the sphere around the joint associated to that marker. There are two intersection points, the one retained is the one whose position is closest to the marker's position in the previous 3D frame. (b) Camera ray in 3D scene, yielding the reconstruction of an elbow marker.

4.2.5.4 Identification by closest-limb and likelihood estimation

If all markers are still not found, we attempt to identify the 3D markers that are still anonymous. We find all the skeleton joints that are missing one or more markers. Assuming that displacement is minimal from one frame to another, we retrieve the co-ordinates of these joints in the previous frame, and calculate the distance from these joints to each remaining unidentified 3D marker (see Figure 4.9(a)). The distance closest to the marker-to-joint distance specified by the marker model yields an association of the 3D marker to that joint. We verify that this



(a) (b) (c)

Figure 4.9: (a) Identification of markers whose marked inventory is incomplete; (b) finding the closest neighbour for each unidentified marker; (c) illustration of the segment rigidity and flexible marker-to-joint distance constraints.

association is plausible by using a 2D and 3D tracking consistency constraint. In other words, if a marker is supposedly identified as being marker *i* in the current frame [*f*], then it is linked in a trajectory to marker *i* in frame [*f-1*]. The set of 2D markers associated to the 3D markers, in [*f-1*] and [*f*], should logically also be linked in trajectories.

We make sure that this is the case, as discrepancies would indicate that the association is erroneous. This is based on the fact that statistically, 2D tracking is more trust-worthy than marker-to-joint association namely due to the fact that 2D positions are reliable and that there is continuity from one frame to the next.

Before accepting the association, another constraint is applied, namely on segment rigidity. The skeleton being an articulated structure with rigid segments, the distance between two joints on a same segment is logically always the same. Therefore, we can extend this fact to the markers associated to the joints. Given that the markers are positioned on a sphere around the joint, the distance from one marker to another is not fixed and rigid, but it is basically contained within an interval whose minimum value is the length of the underlying bone segment $bone_{length}$ minus the two marker-to-joint distances d_1 and d_2, and whose maximum is the length of the segment plus the two distances (the minimum possible distance between the two markers is $[bone_{length} - d_1 - d_2]$ and the maximum distance is $[bone_{length} + d_1 + d_2]$ at the extremes of the spheres). We check that these distances match for all markers on a same segment, for the associations found, and if there is a none-match, the association is rejected. See Figure 4.9(b) for an illustration of this constraint.

Combining constraints on rigid segment lengths and flexible marker-to-joint distances, and applying a closest-neighbour matching between markers and skeleton, we have ensured the robustness of the system with respect to artefact motion of the markers. Indeed, marker-to-marker and marker-to-joint distances can vary without causing a breakdown of the algorithm, due to the fact that these are not eliminatory conditions for marker identification.

4.2.5.5 Position correction

If the distance from marker to joint is larger than the distance specified by the marker model, we "bind" the co-ordinates of the 3D marker to the joint, modifying its 3D co-ordinates so that the marker moves within an acceptable distance of the joint. We however leave all reliably reconstructed 3D markers untouched. This process is reliable if the variance of the measured marker-to-joint distance was not too large and if sufficient data was provided in order to calculate the marker-to-joint distances.

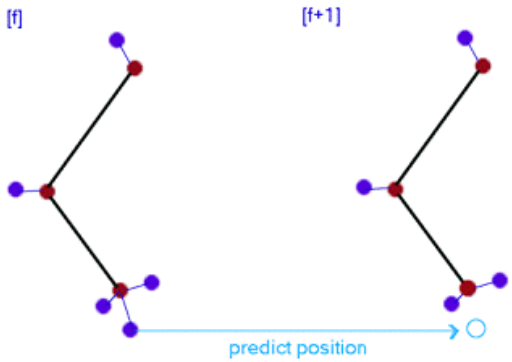


Figure 4.10: Prediction of a marker position in the next frame, on the basis of its position in the previous frame and of its acceleration over a four-frame sliding window.

4.2.5.6 Marker inference

In the case where after all these steps, we still do not have the complete number of markers, we need to estimate their positions and create them using a prediction scheme in a four-frame sliding window. If a given joint is missing markers, we retrieve the positions of the latter in the three previous frames [*f-3*], [*f-2*] and [*f-1*], and calculate the acceleration. We apply this acceleration to the position in [*f-1*], thus obtaining an estimated position of the marker in the current frame [*f*] (see Figure 4.10). As before, we calculate the distance from this inferred position to its associated joint. If it is out of range, we "bind" the co-ordinates, the binding distance being proportional to the computed marker-to-joint distance. In this manner, all 3D markers are available for the pose recovery process, carried out by least-squares fitting.

4.3 Experimental validation

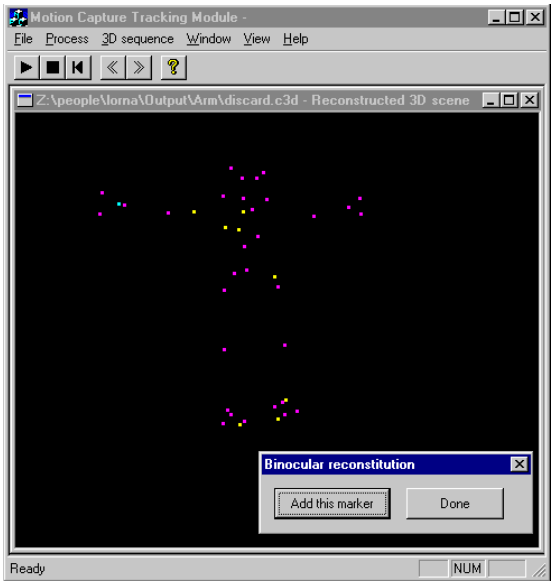
In the previous section, we introduced our approach, here we present the complete work-flow from input to output, and the various steps involved.

4.3.1 Motion capture input data

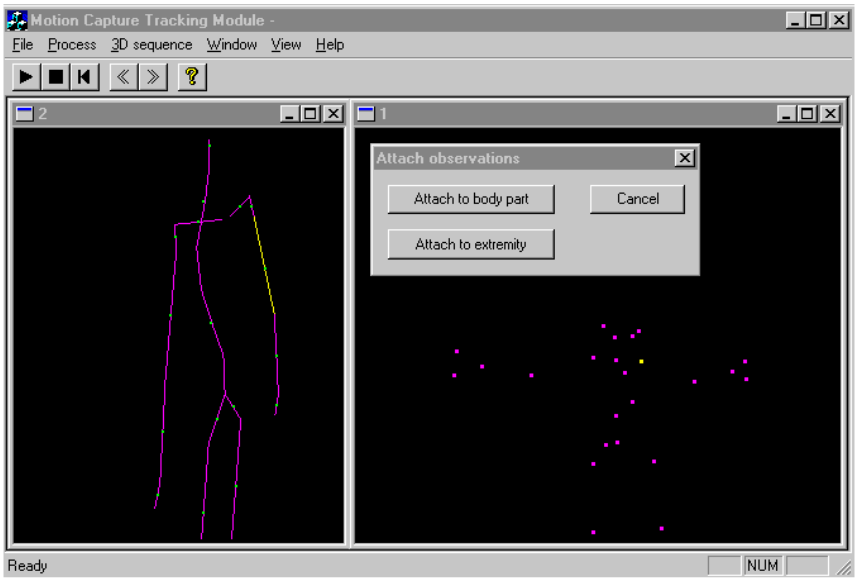
The 2D camera data and calibration parameters were provided by an Elite^{TM} optical motion capture system [Ferrigno and Pedotti1985]. The subject carrying the reflective marker is imaged by eight infrared cameras, the Elite system returning a 2D location for each visible marker. We will hence be working on sets of 2D point locations, one for each marker and each camera that sees it, and a projection matrix for each camera.

To extract a 3D animation of a skeleton from a variety of movements performed by the same subject wearing the same marker set, we first need to derive the body-and-marker model, that is a body model scaled to the match the actual size of the subject and an estimate of the locations of the markers with respect to the joints. To achieve this result, the subject is asked to perform a "calibration motion," that is, a sequence of simple movements that involve all the major body joints. We can then use this calibrated skeleton for further motion capture sessions of more complex motions.

We need to start - as for each captured motion - by reconstructing the markers in 3D. As the calibration motion is an especially simple routine highlighting the major joint motions, the 3D location of the markers can be automatically and reliably reconstructed without knowledge of the skeleton for 200 to 300 successive frames. In practice, we partition the calibration motion into independent sequences, each one involving only the motion of one limb or body part at a time. We then perform 3D reconstruction and tracking for each one independently. If necessary, the user can re-attach some markers to specific body part if they become lost.



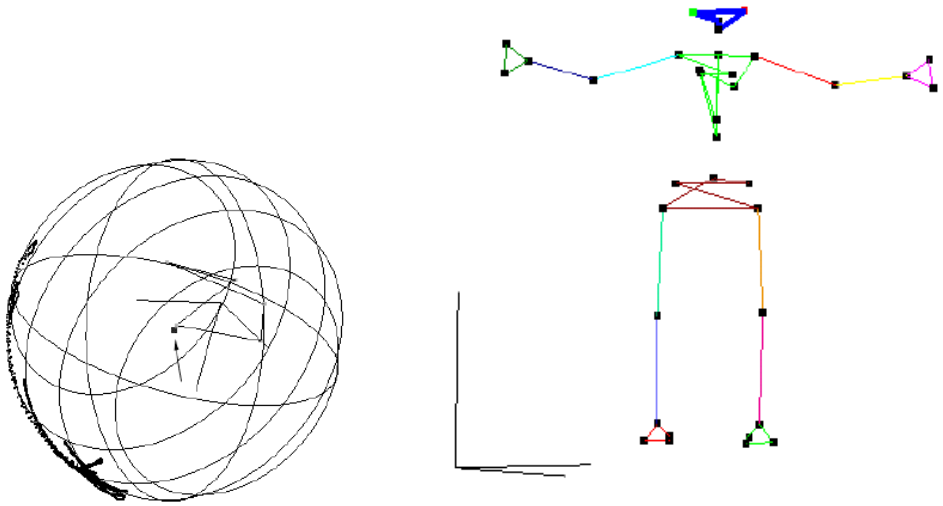
(a)



(b)

Figure 4.11: (a) User confirmation for 3D markers reconstruction using binocular vision; (b) manual marker to body part association, attaching a marker either to the joint or to the extremity of the limb.

In the first frame of the sequence, 3D reconstruction using trinocular stereo matching is followed by a binocular pass, in order to attempt to reconstruct the missing markers. Binocular stereo matching is bound to introduce errors so at this stage, the user is prompted to confirm whether or not these binocular reconstructions are correct,



(a) (b)

Figure 4.12: (a) Estimating the centre of rotation of a marker around a segment; (b) maximal partitions obtained by body model segmentation [Silaghi *et al.*1998].

as any error is bound to be propagated throughout the process. As soon as all markers are found in the first frame, the user is asked to associate each marker to a joint. For each highlighted marker, the user must select a body part and corresponding joint. Any marker not associated to a body part is discarded during the fitting process. Such a user intervention step could theoretically be avoided if a standard initialisation pose was defined, but this would decrease the flexibility of the system in terms of body motion structure and marker positioning.

Once these associations have been manually carried out, we can proceed with 2D and 3D tracking of the markers, thus giving us the calibration motion reconstructed in 3D, the trajectories of the markers throughout the sequence, as well as the identification of the markers with respect to the skeleton model. These two only user interventions are shown in Figure 4.11(a) and (b) respectively.

4.3.2 Acquiring the body-and-marker model

During the calibration phase, our goal is to scale the bones of the generic body model so that it conforms to the measurements of the subject, as well as to model the locations of the marker with respect to the joints. The body-and-marker model is computed by least-squares minimisation (see the Appendix, Section 9.3). As this is a non-linear process, the system goes through three successive adjustment steps so as to move closer and closer to the solution at an acceptable cost while avoiding local minima. These steps are described below.

4.3.2.1 Initial joint localisation

A non-iterative technique was developed in [Silaghi *et al.*1998] allowing us to use the tracked markers to roughly estimate the 3D location of a few key joints in each frame of the sequence, as well as the relative 3D trajectories of the markers with respect to the underlying joints. We describe this technique briefly below.

Let us consider a referential bound to a bone represented as a segment. Under the assumption that the distance between markers and joints remains constant, the markers that are attached on adjacent segments move on a

Human Anthropometric Data

Segment	Definition	Segment Wt/ Total Body Wt	Centre of Mass / Segment length	Centre of Mass / Segment length	Radius of Gyration / Segment length
			Proximal	*Distal*	*C of G*
Hand [see also]	wrist/knuckle II digit 3	.006	.506	.494	.297
Forearm	elbow/ulnar styloid	.016	.430	.570	.303
Upper arm	G.H jt/elbow	.028	.436	.564	.322
F'arm+hand	elbow/ulnar styloid	.022	.682	.318	.468

Figure 4.13: Excerpt of an anthropometric data table.

sphere centred on the joint that links the two segments, as shown in Figure 4.12(a). The position of a segment in space is completely defined by three points. Thus, if we have a minimum of three markers on a segment, we can define the position and orientation of that segment in space. Afterwards, we compute the movement of the markers on adjacent segments in the referential established by these markers and estimate their centres of rotation.

To take advantage of this observation, we partition the markers into sets that appear to move rigidly and estimate the 3D location of the centre of rotation between adjacent subsets, which corresponds to the joint location. This is illustrated in Figure 4.12(b).

This yields the approximate 3D location throughout the sequence of thirteen major joints, namely the joints of the arms and legs, as well as the location of the pelvic joint, at the base of the spine.

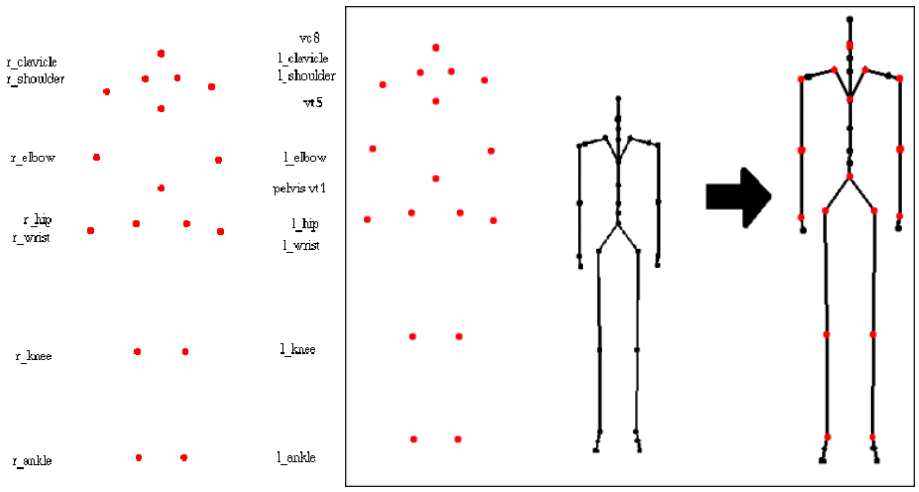
4.3.2.2 Body model initialisation

Given these thirteen joint locations, we take the median distances between them to be estimates of the limb lengths of the subject. We then use anthropometric tables to infer the length of the other body model segments, such as the table shown in Figure 4.13 extracted from [Badler *et al.*1993]. This gives us a static stick figure model scaled to the size of the subject, meaning it has the appropriate dimensions but does not yet capture the postures for the calibration sequence or the relative position of markers and joints.

To estimate those marker-to-joint distances, we first need to roughly position the body model in each frame by minimising the distance of the thirteen key joints to the corresponding centres of rotation. This is done by minimising an objective function that is the sum of square distances from the centres of rotation to their corresponding joints. Fitting is performed frame-by-frame, and the initial parameter values for frame [$\boldsymbol{f}$] are the optimised parameters obtained from the fitting in the previous frame [$\boldsymbol{f\text{-}1}$]. As we only have thirteen observations for each frame, we do not attempt to estimate all of the model's DOFs, as minimisation would inevitably fail in this case. Only ten joints (shoulders, elbows, hips, knees, pelvic joint and the fourth spine vertebra) are active while all the others remain frozen. This yields the postures of the model in all frames of the calibration motion. In other words, we now have values of the global positioning vectors and degrees of freedom in each frame, as well as a better approximation to the limb lengths of the model, as shown in Figure 4.14(b).

Least-squares sense fitting is applied using the joint positions of the local fitting as observations, and the stick figure as the fitting model, in other words, we will modify the parameters of the model in order to minimise the objective function, i.e. the distance d from the observations to the model.

The least-squares sense minimisation scheme stipulates that the function to be minimised is the sum of the



(a) (b)

Figure 4.14: Computation of the joint positions for all frames of the calibration motion: (a) the 13 major joints; (b) scaling of the body model and least-squares fitting of the joint rotation centres to the actual model joints.

squares of the errors. For a given frame, the error of the fitting is the current distance from the observation to the model. In the absence of noise, this distance should ideally be zero.

The joint positions calculated by the local fitting module are in world-space co-ordinates. In order to measure the distance from the observation to the model, we transform the joint positions from global world co-ordinates to the local joint co-ordinates. Once this transformation has been applied, the distance is simply the norm of the observation (as the joint of the model is now the referential at co-ordinates (0, 0, 0)). The square sum of errors should be zero, as we are fitting approximated joint positions to the optimised joint positions of the model.

The square sum of errors is:

$$\sum_{f=0}^{N} ||obs_{local}||^2 = 0$$

where ***N*** is the total number of frames, $||obs_{local}|| = \sqrt{x^2 + y^2 + z^2}$ is the objective function to minimise, ***(x, y, z)*** being the local joint co-ordinates of the observation.

For each frame, the error is:

$$f_{obj} = \sum_{i=0}^{M} (x_i^2 + y_i^2 + z_i^2)$$

for a model with ***M*** joints.

The parameters of the model, corresponding to a human body skeleton representation, that are to be adjusted are the following:

- the bone lengths
- the global position in space of the model (x, y, z of the translation vector, and α, β, γ of the rotation vector)
- the degrees of freedom (DOF) of the joints, each joint having from one to three DOF.

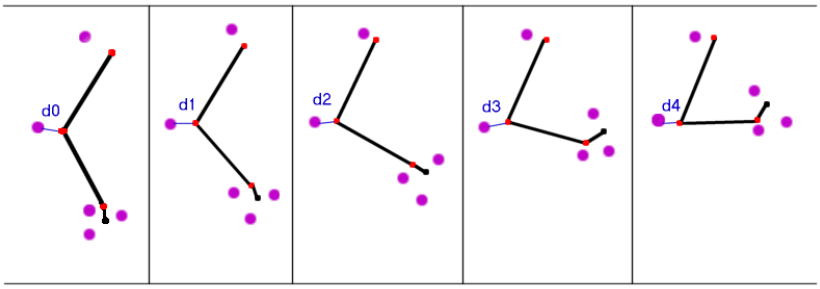


Figure 4.15: Initial median marker-to-joint distance estimate; for each of the five frames, we have the reconstructed markers, the estimated posture, and we measure the distance from each marker to its associated joint, typically for the elbow marker, we will calculate the median marker-to-joint distance of $(d_0, d_1, d_2, .., d_N)$.

In order for the least-square fitting to converge towards an optimal value for each parameter, the number of parameters to solve must not be too high with respect to the number of observations (between 30 and 40, typically), otherwise the system would be under-constrained. This is the reason why we have simplified the model in order for the model parameters to be reduced to the following:

- 32 degrees of freedom (15 joints)
- 14 bone lengths
- 6 parameters for setting the global position of the model in 3D space.

Minimising the objective function means determining the zeros of its partial derivatives with respect to all model parameters θ, and these can be computed using the chain rule:

$$\frac{\partial f_{obj}}{\partial \theta} = \frac{\partial f_{obj}}{\partial obs_{local}} . \frac{\partial obs_{local}}{\partial \theta}$$

The derivatives of the second half of the equation are calculated in Maple and exported as C code. As to the first partial derivatives, they are calculated are those with respect to the parameters $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ (i.e. the position of the parent joint of the associated body part) :

$$\frac{\partial f_{obj}}{\partial obs_{local}} = \left[\frac{\partial f_{obj}}{\partial x}, \frac{\partial f_{obj}}{\partial y}, \frac{\partial f_{obj}}{\partial z} \right]$$

where

$$\frac{\partial f_{obj}}{\partial x} = \frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{\partial f_{obj}}{\partial y} = \frac{y}{\sqrt{x^2+y^2+z^2}} and \frac{\partial f_{obj}}{\partial z} = \frac{z}{\sqrt{x^2+y^2+z^2}}$$

The co-ordinates of the joint are themselves function of the global position and degrees of freedom of the joints preceding it in the hierarchy.

For each frame, and each observation, we compute the value of the derivatives of the objective function (one for each body part), and adjust the parameters of the model in order to minimise the objective function. Three to five iterations usually suffice to yield accurate estimations.

As the joint positions obtained by local fitting are all identified, we know to which body part each observation belongs, and to which joint of the model it should be fitted, so from here on, the least-squares fitting process can run un-hindered. As a result, we obtain the posture of the stick figure model in all frames of the calibration motion, these postures then serving the purpose of initial values of the model parameters for the next fitting stage. The closer the initial postures are to the solution, the less iterations will be necessary at the next step.

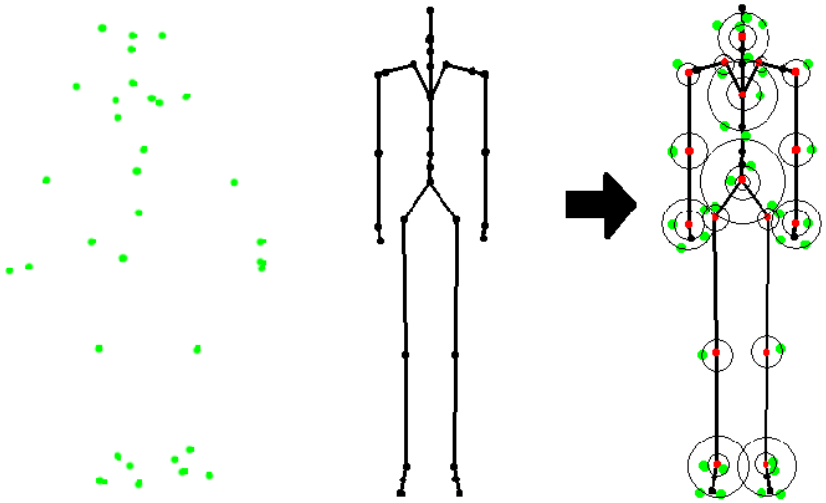


Figure 4.16: Fitting of a model to reconstructed 3D markers using the previously computed marker-to-joint median distance as objective function value.

4.3.3 Global fitting

We now have a model that is scaled to the size of the performing subject, but we are still missing a complete marker model, meaning one that specifies where the markers are positioned on the body of the subject and their distance to the joints they are attached to. This is computed by performing a second least-squares minimisation pass, where the new observations are the actual 3D marker locations to which we will fit the model. Markers not being located exactly on the joints, the marker-to-joint distances must be estimated. To this end, we superimpose the 3D marker co-ordinates with the previously computed postures. In each frame, we then compute the distance from the marker to the joint and we take the median value of these distances to be our initial estimate of the marker-to-joint distance (see Figure 4.15).

When fitting our model to the 3D markers, the value of the objective function to attain for each model joint and observations is this median distance (see Figure 4.16):

$$\sum_{f=0}^{N} ||obs_{local}||^2 = \overline{d}$$

where obs_{local} are the co-ordinates of the observation in the local joint referential and d is the median observation-to-joint distance. Taking the marker model to be the distance from marker to joint means that the marker is expected to always be located on a sphere centred in the joint.

We now have all the information required to fit the model to the observation data. The initial state of the model in each frame is given by the previously obtained skeleton postures, and all markers are identified by the user in the first frame of the sequence.

As we need to check that all markers are present and identified before fitting, we do it one frame at a time. Technically, the fitting process is similar to the one we used to fit models to stereo video sequences [Plänkers and Fua2001]. The interested reader is referred to [D'Apuzzo *et al.*2000] for details on the algorithm.

For each frame and for each marker, once the fitting is complete, the distance between marker and joint is stored. At the end of the calibration motion sequence, we have as many such distances per marker as there are frames. The median value of these distances is an improved approximation of the marker-to-joint distance and becomes the final marker model. The complete process from top to bottom is shown in Figure 4.17.

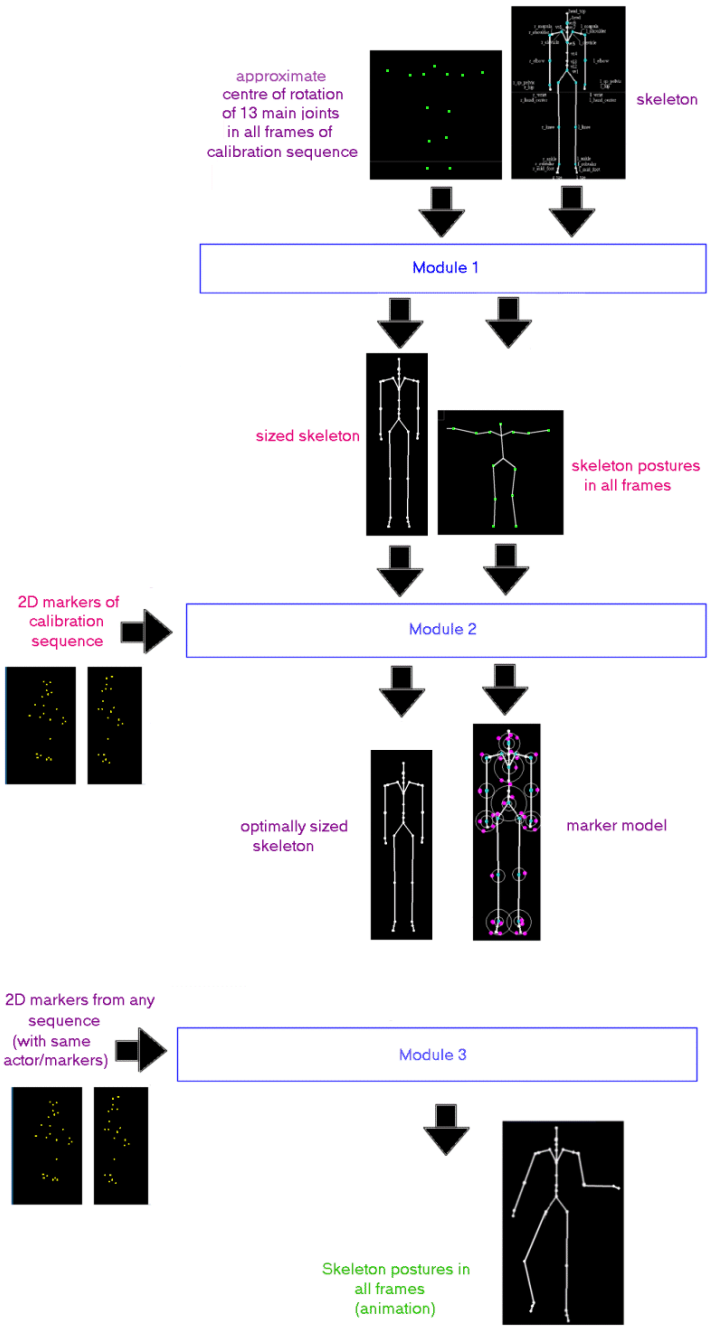


Figure 4.17: Summary of skeleton-based tracking and fitting modules.

4.3.4 Capturing Complex Motion

The resulting body-and-marker model can now be applied to motions that we actually wish to capture. The procedure is identical to the one used in the global fitting step of the previous section, with the difference that the user is now required to identify the reconstructed 3D markers in the first frame by associating them directly to

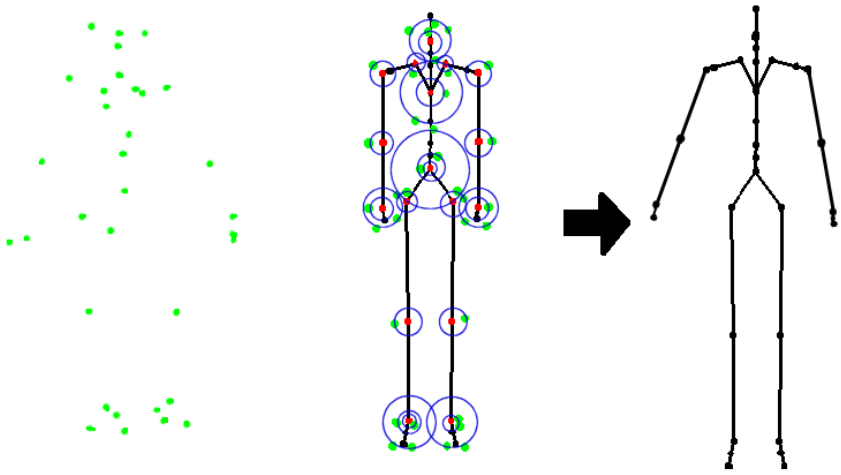


Figure 4.18: Fitting of 3D reconstructed markers to a body-and-marker model, to recover the corresponding posture parameters.

3D markers located on the body-and-marker model, as per the model computed during the calibration phase. The entire fitting procedure is then carried out, by minimising the distances between the reconstructed 3D markers and the markers of the previously-computed model, thus yielding a posture in each frame (see Figure 4.18).

The images in Figure 4.19 show a few results obtained with some of the movements provided by the motion capture studio. The use of the skeleton model has enabled us to improve every step of the process, from 3D reconstruction, to tracking and identification of the markers. It is robust with respect to noisy data, out-of-bound and non-identified markers will be rejected, and it can also handle the case of occluded markers.

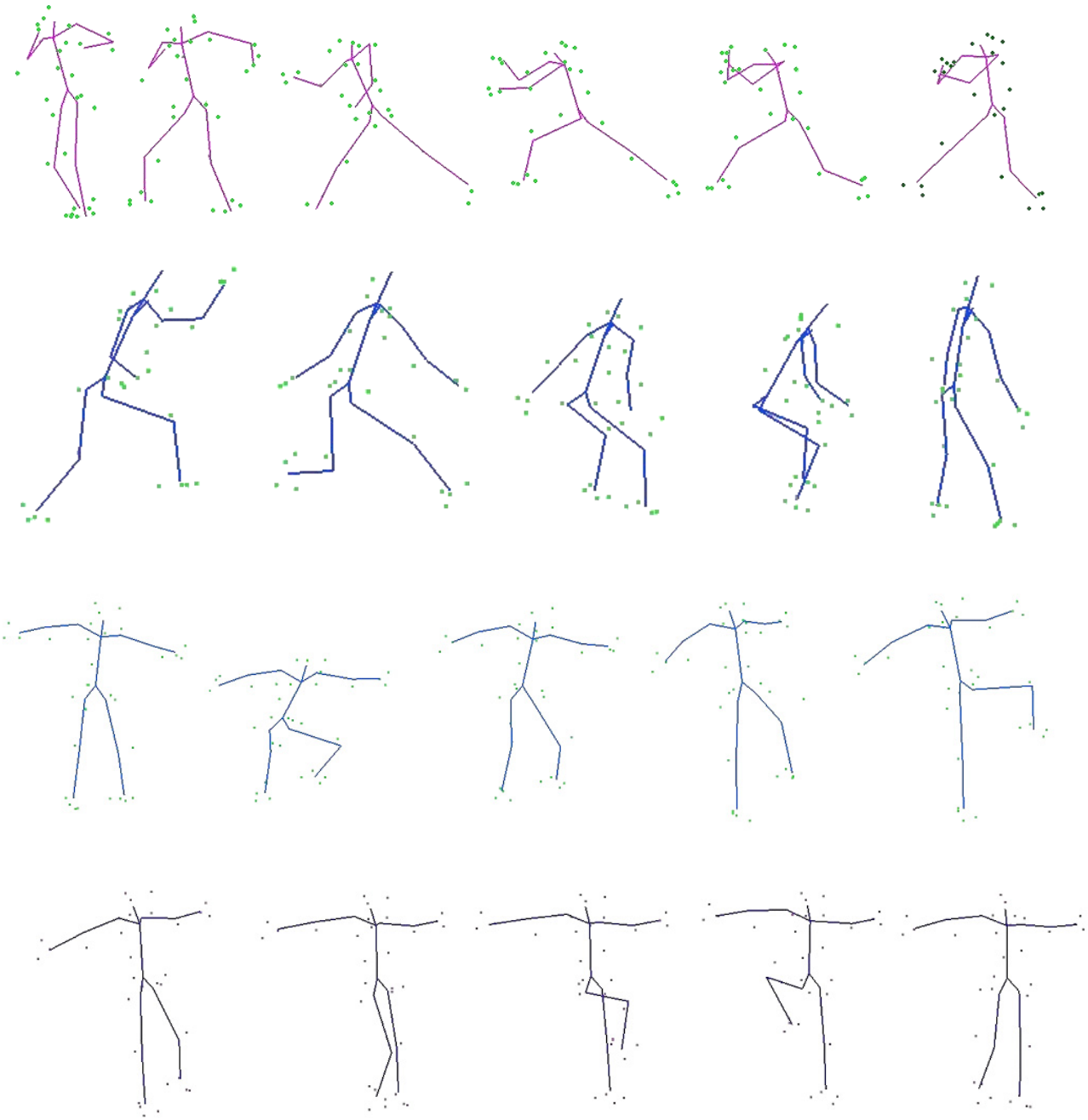


Figure 4.19: (1) Karate motion; (2) jump motion; (c) knee joint and (d) hip joint motion in the calibration sequence.

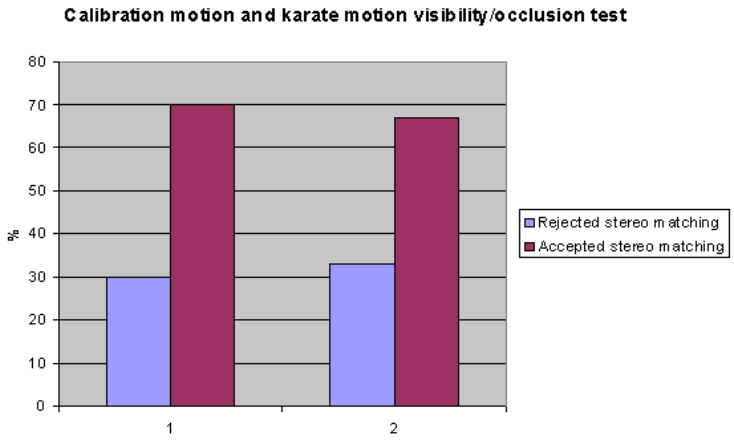


Figure 4.20: Visibility and occlusion test statistics over 400 frames.

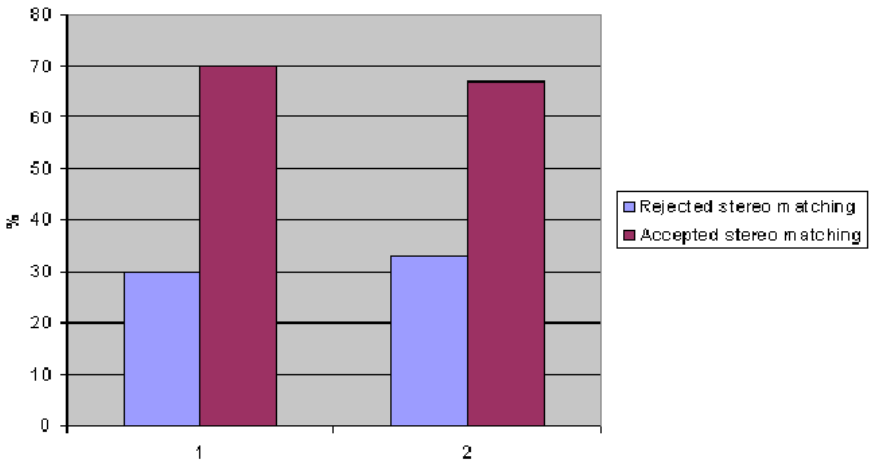


Figure 4.21: Visibility and occlusion test statistics over 800 frames of the gym motion, one trial without the test, and one with.

4.4 Outcomes

4.4.1 Improvements provided by the skeleton-based tracking

Here we attempt to quantify the improvements brought about by using the model with respect to marker reconstruction and tracking. In Figure 4.20, we present some statistics on the improvement resulting from the visibility and occlusion test with respect to accurate marker reconstruction. We used 400 frames of, respectively, the calibration and karate motion. We show the percentage of marker pairs that were rejected/accepted by the visibility/occlusion constraint. These statistics underline the fact that one third of the stereo correspondences were doubtful, and a better match was to be sought.

In Figure 4.21, we compare the number of correct stereo matching associations found by the algorithm with or without the visibility/occlusion constraint. Our reference for this test was a copy of the same 2D file used for input, but with the markers having previously been manually identified (file provided by the motion capture studio). Note that the test produces a higher percentage of correct stereo matches than simple reconstruction.

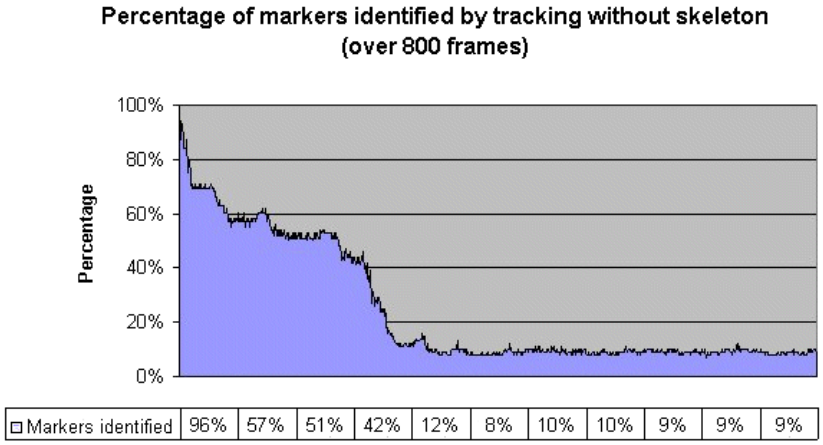


Figure 4.22: Percentage of markers identified by simple tracking, for the calibration motion.

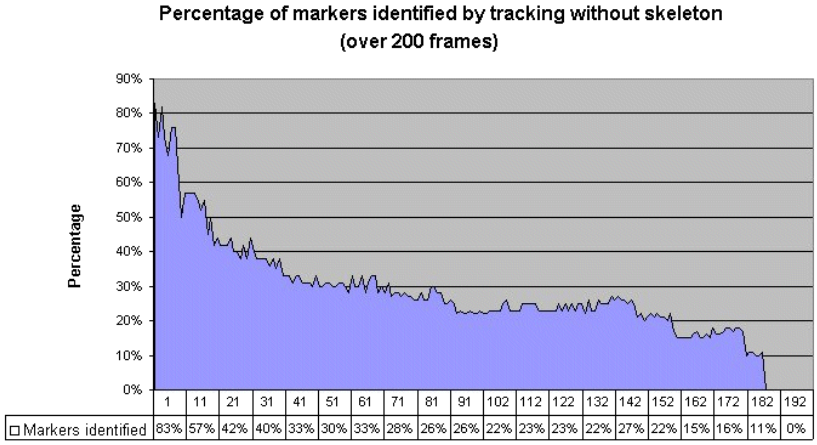


Figure 4.23: Percentage of markers identified by simple tracking, for a captured karate motion.

The percentage gained may seem small, but we need to keep in mind the fact that errors are propagated through 3D tracking and will therefore affect the entire process.

In Figure 4.22, we have run 3D tracking over 800 frames of the calibration motion. This tracking uses only simple marker prediction without body model information over a sliding window of four frames. If a marker is lost in a frame, it cannot easily be recovered in any of the subsequent frames, unless the subject remains immobile. The figure shows that after about 300 frames, the number of tracked markers drops to about 10% (tests performed with a subject using 32 body markers). In Figure 4.23, we have performed the same statistics, but this time using a captured motion containing a fast movement, and we notice that simple tracking loses all markers in less than 200 frames. Note that when we say that a marker is 'lost', this means that a complete trajectory from frame zero to the present frame is no longer available. The marker can still be tracked to a marker in the previous or following frames, but the trajectory will be fractured there where the marker was not identified.

With skeleton-based tracking, all markers are recovered in all frames, i.e. 100% of the markers are present and identified, keeping in mind the fact that a certain percentage of these markers were reconstructed by prediction, their co-ordinates thus being subject to a certain error factor.

In Figure 4.24, we show the percentage of markers identified by each process of the marker inventory of

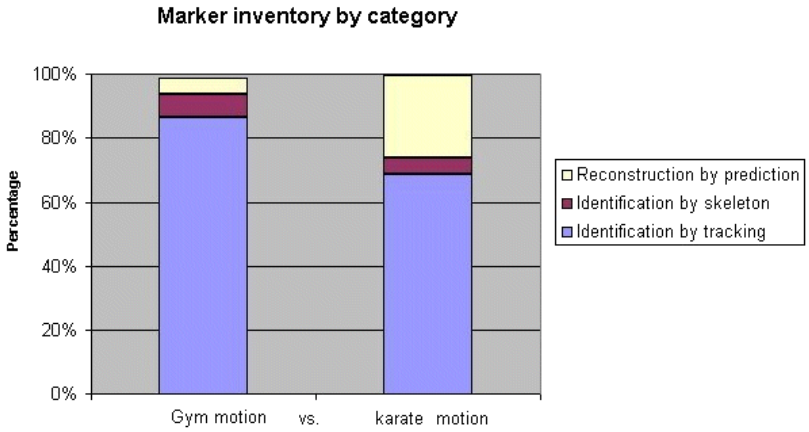


Figure 4.24: Percentage of markers identified by each step of the marker inventory, respectively for the calibration and karate motion.

skeleton-based tracking. The two bar charts compare these percentages in the cases of respectively the calibration and karate motion. The first step of the process identifies reconstructed markers using simple 3D tracking. This percentage is obviously higher than in the previous figures, because in the skeleton-based tracking case, all markers are present in each frame, whereas in the simple tracking case, a marker lost in one frame is lost forever. The second step identifies reconstructed markers using the position of the model in the previous frame. As to the third step, it reconstructs the markers that are still missing in the current frame, combining prediction of the 3D tracking type and the position of the underlying body model. The obvious observation would be that identification by skeleton proximity and reconstruction by marker prediction have little influence, with respect to the performance of tracking. However, the reality is not that straightforward, if we keep in mind that a broken tracking link is not easily recovered in an accurate manner, no matter how efficient the tracking algorithm is. Therefore, full marker reconstruction and correct identification are absolutely essential; as shown by Figures 4.22 and 4.23, the identification rate would soon drop to zero without it.

The obtained output DOF values for hip abduction and flexion, and knee and elbow flexion are shown in Figure 4.25. In Figure 4.26, we suppose the right leg joint angle values onto the marker reconstruction and identification statistics so as to be able to determine whether any particular errors are introduced by these methods. The peaks represent integer values of respectively 0, 1 or 2, in the cases where the markers are respectively identified through "Identification by tracking" (paragraph 4.2.5.2), "Identification by closest-limb and likelihood estimation" (paragraph 4.2.5.4), and "Marker position prediction" (paragraph 4.2.5.6), for 400 frames of the karate motion. The figures show that following a marker on the knee is relatively easy and that tracking can cope with that single-handedly. However, for the more complex case of the hip joints, simple tracking is insufficient after 100 frames, and the alternative methods take over, without introducing gross errors.

4.4.2 Fitting results

The images in Figure 4.19 show a few results obtained with one of the movements provided by the motion capture studio. The use of the stick figure model has enabled us to improve every step of the process, from 3D reconstruction, to tracking and marker segmentation. It is robust with respect to noisy data, out-of-bound and non-identified markers will be rejected, and it can handle the case of occluded markers.

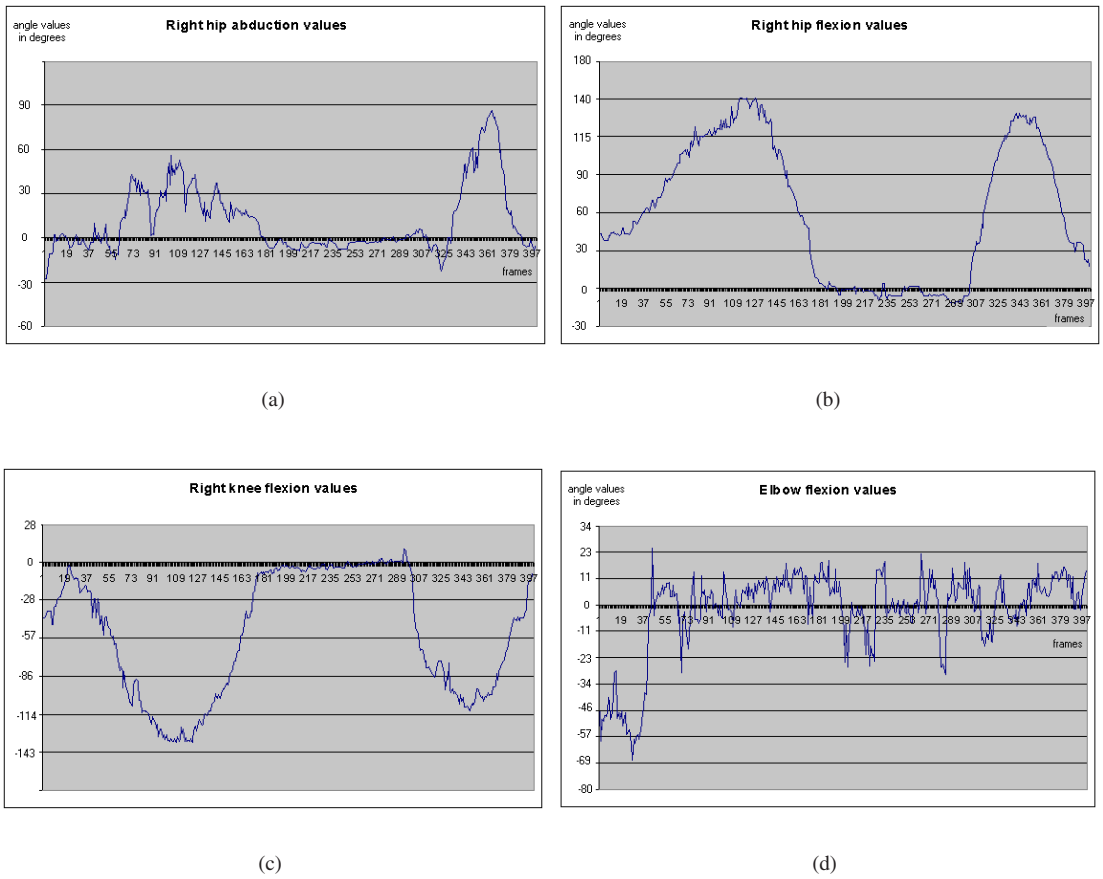


Figure 4.25: Rotational parameter graphs for (a) hip abduction; (b) hip flexion; (c) knee flexion and (d) elbow flexion, over 400 frames.

4.4.3 Discussion

In this category, we applied spatial and temporal coherence constraints to the specific case of optical motion capture, during which we derived a simple distance function relating the markers to the underlying model joints. The purpose of the latter was to aid marker identification before posture recovery and eliminate eventual segmentation errors. The initial identification was carried out through tracking. In each respect, we could suggest some improvements.

4.4.3.1 Data model

Over the motion capture process, we observed what the typical distance from each marker to its underlying joint was. We then assumed that the relative motion of a marker with respect to the joint was located on the surface of a sphere centred in the joint, the computed distance being its radius. Such a simple constraint greatly improves the segmentation process, but could be even more reliable if the marker motion model were more close to reality. For this, we could for example use the physical location of the limb to eliminate a good portion of the sphere on which the marker could not possibly be located.

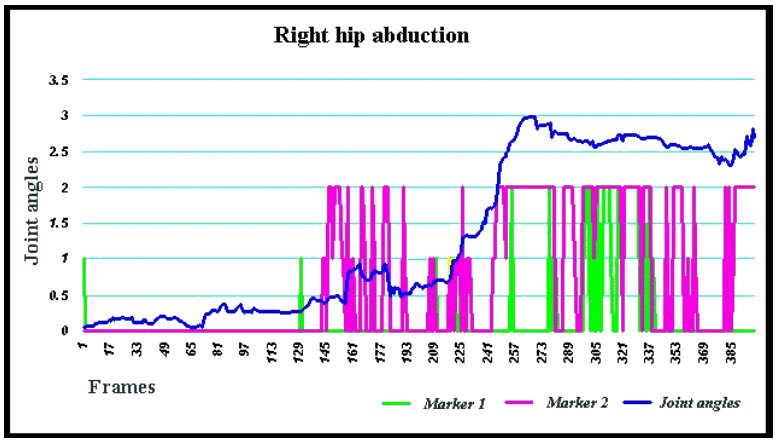
4.4.3.2 Temporal coherence constraints

In the four-frame tracking scheme we apply for determining marker trajectories, we define search neighbourhoods whose radius is constrained to be smaller than given velocity and acceleration maximal values. The efficiency of the tracking could certainly be improved if these radii could be defined dynamically, for example on the currently observed speed of motion.

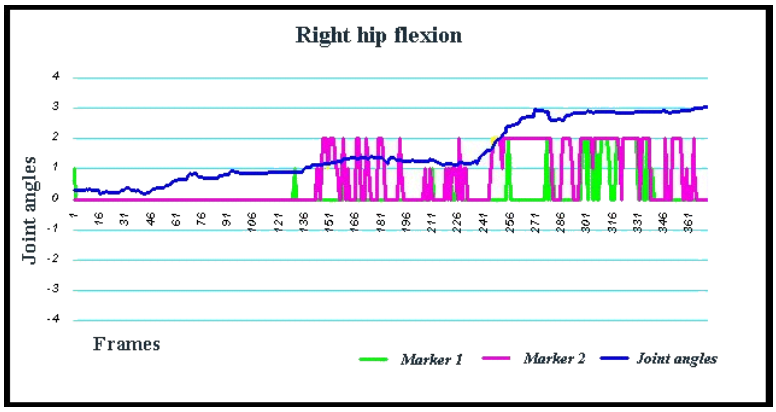
4.4.3.3 Problem over-determination

Given the simplicity of our human body model, and the sparsity of the data at the extremities, we are unable to infer correct orientations for them. This is due to the over-determination of the system, meaning that there is no single solution to the problem. For example, determining forearm orientation on the basis of one observation on the elbow and one on the wrist is impossible, as any amount of twist at the shoulder, elbow or wrist level would equally correspond to the observations and therefore be a valid solution. If this is not immediately obvious from recovered motion on a stick figure, it would be if we applied our recovered motion parameters to a multi-layered virtual human in an animation package.

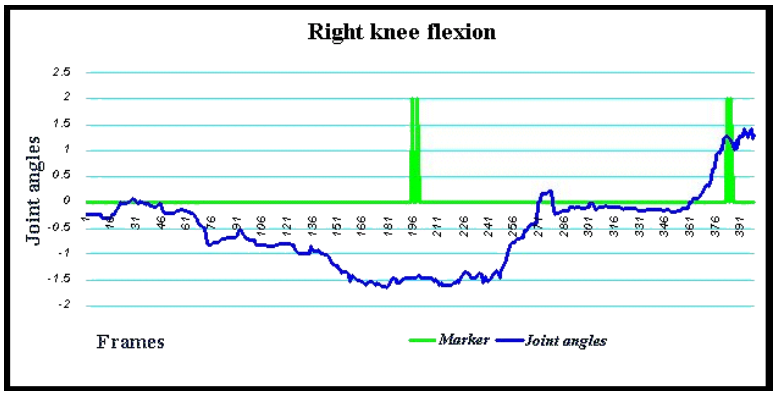
These problems could be overcome either by adding markers to the marker set, so as to be able to infer correct extremity rotations, or by encoding the natural physical limitations of the human body within the model itself. This is the turf of the following chapter, which deals with what we refer to as intrinsic constraints, namely constraints that are encoded within the body model.



(a)



(b)



(c)

Figure 4.26: Graphs of the optimised (a)(b) hip and (c) knee DOF values over 400 frames, including statistics on marker identification.

Chapter 5

Imposing intrinsic motion capture constraints

In the previous chapter, we defined the notion of "plausible" conditions related to all constraints applied independently of the body model. Such conditions were referred to as extrinsic constraints, whose purpose was to efficiently prune the search space of motion parameter solutions. State-space complexity can be further reduced by applying a second type of constraint, namely of the intrinsic kind. Such constraints consist of information that is incorporated directly into the model itself, this including:

- joint limits that set the range of rotational freedom of the joints;
- collision avoidance, which is enforced by verifying that body parts do not intersect each other;
- equilibrium, gravity etc., these being generally enforced by equations simulating dynamics.

Strangely enough, intrinsic and extrinsic constraints seem to have been used in a mutually exclusive way, as appears from Table 5.1[1]. Where multiple views are available, extrinsic constraints were deemed sufficient, intrinsic constraints having mostly been applied in motion capture from monocular sequences. The notable exceptions to this trend appear to be [Moeslund and Granum2000] and [Demirdjian2003]. One would have thought that the combination of constraints would maximise the chances of retrieving the correct posture!

In this chapter, we will deal with intrinsic constraints encoded within the model itself, thus making them applicable to any kind of motion capture, using single or multiple cameras. We will be illustrating such constraints on a motion capture framework based on multiple CCD cameras where stereo data has previously been extracted. This 3D data will be used as-is for matching to a model, no further information being available for the fitting process. In this way, when applying intrinsic constraints, we will be able to observe the impact of - and only of - the constraints on the posture recovery process.

The type of intrinsic constraints we will be focusing on are joint limits, this choice being motivated by the following facts:

1. a satisfying joint limits representation does not seem to exist, for reasons that we will provide in this chapter;
2. they have rarely been enforced in the context of multiple-camera motion capture.

Collision avoidance and human dynamics, on the other hand, have been relatively extensively studied, in spite of the fact that they have rarely been incorporated into a video motion capture context. Furthermore, collision

[1] The use of extrinsic constraints is not featured in the table as it was naturally used when multiple views were available.

Authors	Multiple views	Intrinsic constraints
Cheung et al.	√	
Chua et al.		√
Demirdjian	√	√
DiFranco et al.		√
Dockstader and Tekalp	√	
Drummond and Cipolla	√	
Gavrila and Davis	√	
Kakadiaris and Metaxas	√	
Kuch and Huang		√
Lee and Kunii		√
Liebowitz and Carlsson	√	
Lin et al.		√
Moeslund and Granum	√	√
Ong and Gong	√	
O'Rourke and Badler		√
Perales and Torres		√
Rehg et al.	√	
Rehg		√
Ren and Xu	√	
Ringer and Lasenby	√	
Segawa et al.		√
Sminchisescu and Triggs		√
Theobalt et al.	√	
Wu and Huang		√
Yonemoto et al.	√	

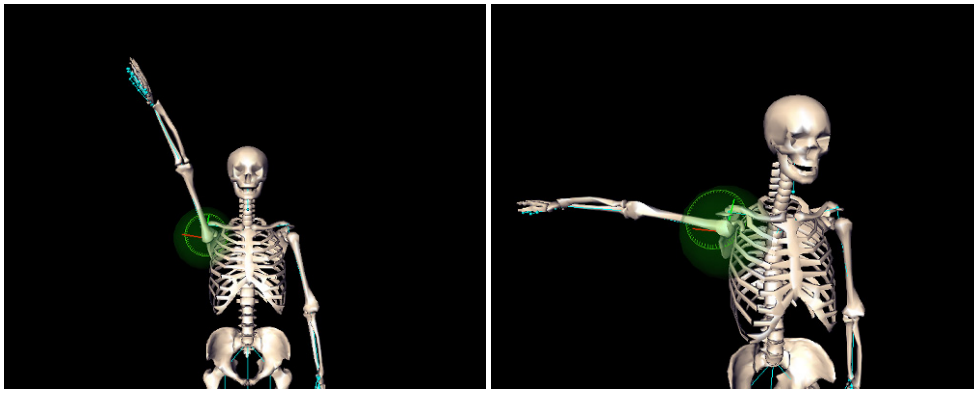
Table 5.1: Use of constraints in motion capture applications.

avoidance is readily enforced by using the shape representation of the body model. For these combined reasons, we will be focusing on joint limits only, in this intrinsic constraint context.

In this chapter, we will start by presenting the particular problem of joint limits in motion capture, then touring the existing formalisms in the literature, and present what we feel are their short-comings. To remedy these, we propose a new representation based on implicit surfaces, which we will apply to a limited area of our human body model, the shoulder and elbow joints, as explained in Section 3.2. The usefulness of the joint limits are demonstrated both for character animation and video-based motion capture.

5.1 Joint limits in motion capture

As can be seen in Table 5.1, a certain number of research groups on video motion capture have included joint limits in their body model, but in our opinion, they do not represent the actual range of motion of the joints, except perhaps in the papers focusing exclusively on the problem of hand posture recovery. This very particular problem being extremely complex due to the juxtaposition of the fingers, their close resemblance in terms of contours, and the great number of occlusions that finger motion involves, the angular constraints on the finger DOFs are in general required to be precise. Nevertheless, the authors have chosen to express the constraints on finger motion in terms of constraint equations rather than actually defining ranges of motion for each rotational degree of freedom. This is understandable as the number of degrees-of-freedom of the finger joints is more often than not limited to one, or at most two, and that the inter-dependencies are relatively simple to express by



(a) (b)

Figure 5.1: Motion inter-dependency between abduction and extension at the gleno-humeral joint: (a) if we perform an abduction of the joint starting with the arm outstretched horizontally at the side, the arm can be elevated almost vertically, the amount of abduction available in this configuration being almost 180 deg; (b) from the same starting position, we extend the arm as much as possible, and then attempt to abduct the arm; sadly, this is impossible: adduction is still possible in such a position but abduction is not, the range of motion of the shoulder joint in terms of abduction/adduction therefore varying as a function of the flexion/extension value.

equations.

In all other cases where joint limits are applied to body joints other than the hand, this has been carried out by minimum/maximum boundary values around each axis of a joint rotation expressed in Euler angles. Regardless of the limitations of such a rotation representation, which we already discussed in Section 3.1.1, defining a range of motion interval on each Euler angle axis does often not reflect reality.

For a 1 DOF joint, where mobility is anyway restricted to rotation around a single axis, setting such an interval is perfectly justified, as such a rotation is continuous. For the cases of 2 or 3 DOF joints, however, such a solution is questionable, for the following reasons:

1. the range of motion in terms of a single axis is not necessarily independent from the motion performed around the other axis (2 DOF) or axes (3 DOF);

2. a joint may also be subject to coupling when other joints in its proximity have a more or less important influence on the its own liberty of movement.

The first problem was already mentioned in Section 3.1.2, when discussing the issue of shoulder joint modelling (Figure 3.11).

To illustrate the second concept with a simple example, we have carried out a simulation on a realistic human skeleton model to which we have applied anatomically-correct joint limits. The shoulder complex is here again the perfect location for our demonstration. In the shoulder model used for these illustrations, shoulder mobility is represented by a single ball-and-socket joint situated at the gleno-humeral joint.

In Figure 5.1, we show the motion inter-dependency that exists between abduction and extension of the shoulder joint (i.e. of the entire shoulder complex moving as a whole). Each motion component (abduction/adduction, flexion/extension and inner/outer axial rotation) represents a rotation around a fixed axis of the local joint coordinate system.

Minimum/maximum boundaries on Euler angles will therefore certainly limit the search space when it comes to posture recovery, but they will do not so in a reliable manner. In general, they can be considered as an upper boundary of the effective range of motion, but their efficiency would be highly increased if they actually reflected reality.

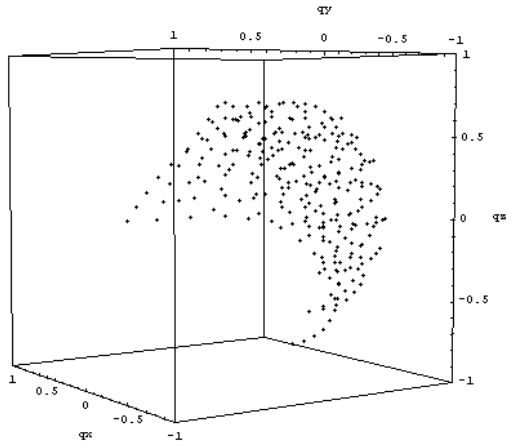


Figure 5.2: Simulation of shoulder compound rotations on the basis of Euler angle min-max values.

5.2 State-of-the-art of joint limits representations

We noted in our state-of-the-art on video-based motion capture that no research work used any joint limits representation other than Euler angle ranges of motion. However, a certain number of more sophisticated formalisms do actually exist in the areas of computer graphics and biomechanics, and a full inventory - at least to the best of our knowledge - is presented in this section.

5.2.1 Boundaries on independent axes of Euler angles

This representation has been debated in the previous paragraph, but we will nevertheless mention it in our state-of-the-art for completeness' sake. Also, we wouldn't want readers, who are incidentally opening this manuscript at exactly this paragraph, to think that we are not aware of this joint limit formalism.

We have already mentioned that it is extremely popular, due to the underlying Euler angles rotation representation, all papers using body models with intrinsic constraints featured in Table 5.1 use such boundaries for their joints.

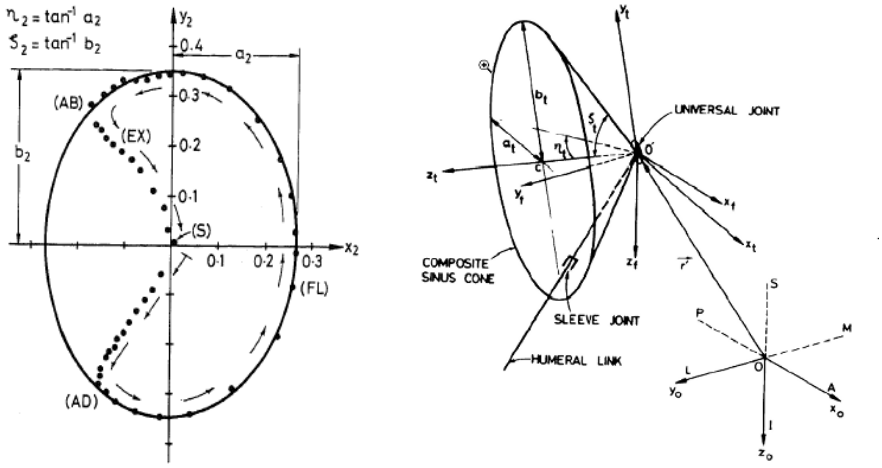
For ulterior comparison purposes, we provide in Figure 5.2 the simulation of the equivalent quaternion rotations[2] of such Euler angle boundaries, for the case of the shoulder compound considered as a single ball-and-socket joint. To create this simulation, the typical minimum and maximum values on the Euler angle rotations were used, and then converted to quaternion rotations. The values are of course not anatomically-correct, they include a fair amount of positions that in reality are not attainable, but they definitely include the totality of feasible rotations. The set $E(i,j,k)$ of Euler angles that is converted into quaternions is defined as:

$$P(i,j,k) = \{p_j, p_j(x_i, y_j, z_k)/0 \leq x_i \leq \frac{\Pi}{2}, -\frac{\Pi}{2} \leq y_j \leq \frac{\Pi}{2}, -\Pi \leq z_k \leq 0\}$$

$$E(i,j,k) = \bigcup P(i,j,k)$$

where the union is performed over a given sampling interval. The Euler angle axis convention used in this case is axial rotation (twist) around the x-axis, abduction/adduction around the y-axis and flexion/extension around the z-axis.

[2] See the Appendix or the following chapter for details on the quaternion notation.



(a) (b)

Figure 5.3: Joint sinus cones of the shoulder complex: (a) cross-section of the joint sinus cone of the gleno-humeral joint [Engin and Tümer1989c]; (b) composite joint sinus cone of the shoulder complex [Engin and Tümer1989b].

5.2.2 Joint sinus cones

The concept of the joint sinus cone was introduced by [Engin and Tümer1989a], defining the notions of "joint sinus" and "joint sinus cone". The first refers to the total range of angular motion of a single joint, and the second, to the conical surface that envelopes the joint sinus and has its apex (the tip) at the functional centre of the joint.

An example of a joint sinus cone and a cone cross-section are shown in Figure 5.3, where the composite joint sinus cone actually represents the range of angular motion of the entire shoulder complex. The cones were constructed on the basis of a statistical database established by [Engin and Chen1986] on the basis of *in vivo* observations of a sample male population aged 18 to 32.

This scheme was recently adopted by [Maurel1998] for 3D biomechanical modelling of the human upper limb joints. The sinus cones of all four shoulder complex joints as defined by Engin and Tümer were modelled, for the purpose of providing realistic animation of an anatomical human body model. An example of the resulting joint boundaries represented as conic shapes with elliptic bases is depicted by Fig. 5.4, including a glimpse of the cone design interface.

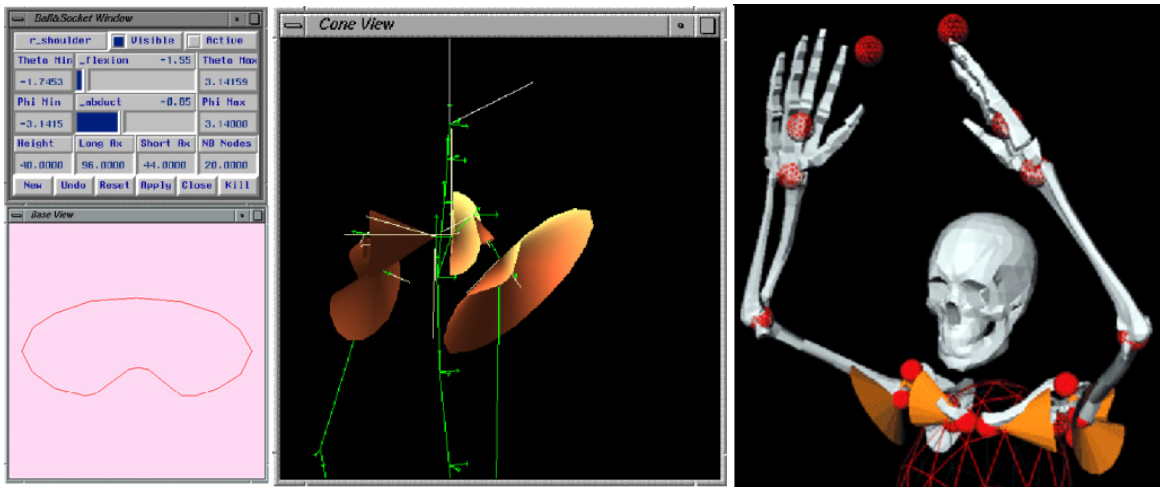
The method presents several undeniable advantages:

- simplicity of the representation,
- independence from choice of rotation formalism,
- simple clamping of an invalid pose to a valid pose by reducing the 3D problem to 2D, and performing an inside/outside test with respect to the polygon representing the cone base.

In spite of the elegance of the method, the reader with a keen eye will have noted that joint sinus cones unfortunately only limit *angular* motion, this leaving one motion component unaccounted for, namely axial rotation.

5.2.3 Spherical polygon

[Korein1985] proposes an alternative formalism for expressing joint limits in the form of a spherical polygon, the principle being similar to the one of the joint sinus cones. This representation reflects the angular motion range of



(a) (b)

Figure 5.4: Joint sinus cones used for computer animation: (a) cone design interface and (b) resulting joint sinus cones applied to animation [Maurel1998].

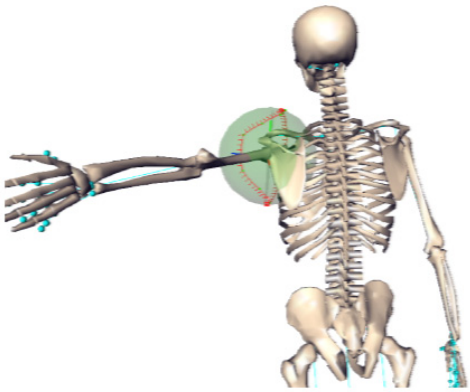
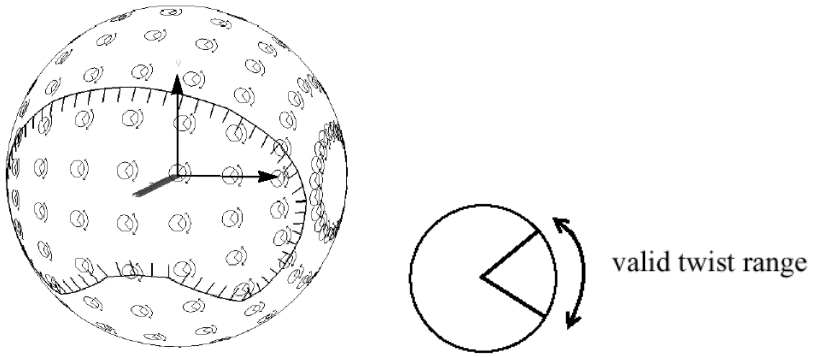


Figure 5.5: Spherical polygon applied to limit shoulder motion.

a spherical joint, where the distal end lies on a sphere centred in the joint. The border of the possible orientations is the joint limit curve, or spherical polygon. An example of such a polygon for the shoulder complex is shown in Figure 5.5.

This representation has been applied in the area of character animation, more precisely for handling inverse kinematics, by [Baerlocher and Boulic2000], and given its similarity with joint sinus cones, it is of course subject to the same limitations, namely that axial rotation is not included. The authors of the previously mentioned paper have designed a scheme in order to circumvent this short-coming, namely by defining local ranges of axial rotation, all over the surface of the sphere, as shown in Figure 5.6. Such a discretised representation is an improvement on the total absence of twist range of motion, but still does not present us with a solution that would enclose all rotation components in a satisfactory manner.



(a) (b)

Figure 5.6: Spherical polygon with local twist, or axial rotation, ranges.

5.2.4 Triangular Bézier patches

A new joint limits model was suggested by [Tolani *et al.*2000], where the workspace of angular motion is defined as a triangular Bézier spline surface (Figure 5.7(a)), on the basis of empirical data, the method having been validated by comparing their output with statistical data available in the literature.

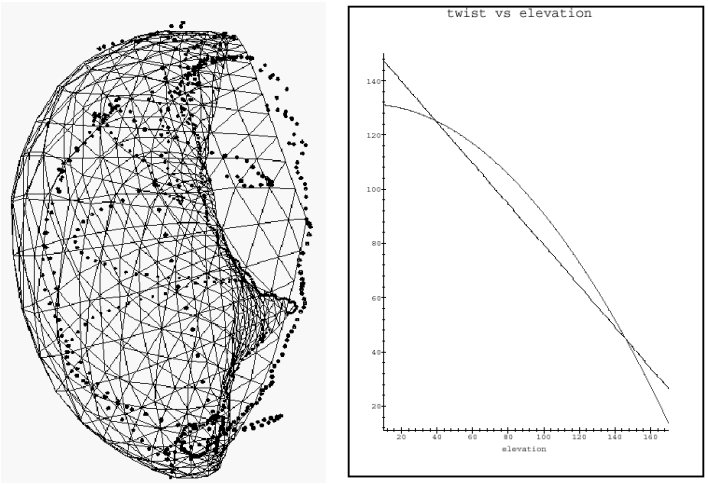
As in the case of the spherical polygon representation, an effort was made in order not to ignore the contribution of axial rotation. To this end, the available range of axial rotation with respect to angular motion was plotted, to which a line or quadratic function was fitted using a least-squares minimiser (Figure 5.7(b)).

Such a representation is similar in its philosophy to the spherical polygon, in the sense that it proposes an explicit formulation for the joint limits, as a set of Bézier patches instead of polygon edges. The problem of motion component separation remains.

5.3 Hierarchical joint limits representation

To summarise the inventory of available joint limits models, most are explicit representations or deal with rotation components separately, or even combine the two, not talking of the Euler angle min/max limits that are completely linked to the rotation representation. On the one hand, explicit representations are computationally costly, and on the other hand, motion component separation ignores the inherent coupling that exists not only between these components but also between proximal joints. It is obvious in the light of what precedes that ideally, a joint limits representation should have the following characteristics:

- measured rotation representation should be independent of the rotation paradigm of the associated joint. If we were measuring Euler angle rotations, the data collected would be totally dependent on the axis convention used and would require additional conversion efforts to be usable;
- the formalism should include all motion components;
- the boundary of the valid motion space should be readily extractable from the collected data and should have an analytical expression, thus yielding an immediate answer to the inside/outside test, as well as being differentiable so that a notion of distance to the boundary is available when determining the closest valid position of a non-valid one;



(a) (b)

Figure 5.7: (a) The triangular Bézier patch representing shoulder compound mobility, the shoulder being considered as a single ball-and-socket joint. (b) Twist is defined as a function of angular motion.

- the representation should be generic and extendible to all joint types, and should allow for inter-joint coupling (as in the case of the shoulder/elbow and hip/knee joints).

In view of the above, we propose a joint limits formalism that is based on data collected from optical motion capture, using a simple measurement protocol involving a small number of markers. From the motion capture data, we infer the corresponding quaternion rotations that can be represented as scattered 3D data if their unitary condition is enforced. The envelope enclosing all the data forms the range of motion of the corresponding joint. This envelope should have a concise mathematical expression, be smooth and continuous, but nevertheless allow detail in its shape, meaning that it can be locally influenced. Implicit surfaces fulfil all the afore-mentioned requirements, and their properties have been applied to the approximation of scattered data representing such complicated shapes as human organs, including detailed items such as arteries [Tsingos *et al.*1995]. Furthermore, by combining implicit surfaces and data voxelisation, we are able to introduce the notion of a sub-space of local joint limits that takes into account the eventual coupling between two joints in the articulated structure hierarchy.

The proposed formalism will be detailed in the following sub-sections, where we will first lay out the main ideas in the context of a single 3 DOF joint. As we have underlined before, dependencies exist not only among the rotation components of a single joint, but also among the rotation components of a set of joints, these joints then being referred to as *coupled*. The extension of our joint limits formulation for a single 3 DOF joint to coupled joints follows immediately from the very nature of our representation. The practical usefulness of our joint limits is subsequently demonstrated not only in terms of motion capture from video sequences, but also in the context of character animation. The more in-depth details of the various steps involved in obtaining such a representation are the topic of the following chapter, this including data collection and processing, followed by boundary point extraction and approximation.

5.3.1 Joint limits for a single 3 DOF joint

Having explained in the previous sections that the greatest difficulty arises when it comes to expressing joint limit boundaries for ball-and-socket joints, we will herewith present a method that does exactly that, and that is transposable to a 2 DOF joint, as it is not restricted to 3D space. For 1 DOF joints, it would obviously be most

impractical to use any other representation that a simple value range.

As we are focusing on the joints composing the human upper limb, we will, in the present sub-section, use the gleno-humeral joint for demonstration purposes. Furthermore, we will be considering it as the sole mobile joint in the shoulder, meaning that the joint limits we will be defining actually limit the range of motion of the entire shoulder complex. The extension to coupled joints, and therefore the joints of the shoulder complex and upper limb in general, will be carried out in a subsequent sub-section.

Our representation consisting of an implicit surface enclosing the set of valid quaternion rotations, we will start by motivating our choice of such a rotation formalism in the next paragraph.

5.3.1.1 Quaternion rotations

Our preference for quaternions over all other rotation representations is motivated by the various advantages provided by quaternions, these being listed for reference in the Appendix.

In practical terms, rotations are represented by the sub-space of unit quaternions S^3 forming a unit sphere in 4-dimensional (4D) space. Any rotation can be associated to a unit quaternion but we need to keep in mind that the unitary condition needs to be ensured at all times. A rotation of θ radians around the unit axis v is described by the quaternion:

$$q = [q_x, q_y, q_z, q_w]^T = [sin(\frac{1}{2}\theta)v, cos(\frac{1}{2}\theta)]^T \tag{5.1}$$

Since we are dealing with unit quaternions, the fourth quaternion component q_w is a dependent variable and can be deduced, up to a sign, from the first three eq.(5.2).

$$q_w = \pm\sqrt{1.0 - q_x^2 - q_y^2 - q_z^2} \tag{5.2}$$

Given collected joint rotation data, by computing the equivalent quaternion rotations, we obtain a cloud of 3D points by keeping the spatial co-ordinates (q_x, q_y, q_z) of each quaternion. In other words, these three numbers serve as the co-ordinates of quaternions expressed as projections on three conventional Cartesian axes. In Figure 5.8 are shown examples of various motions of the shoulder, displayed as 3D quaternions.

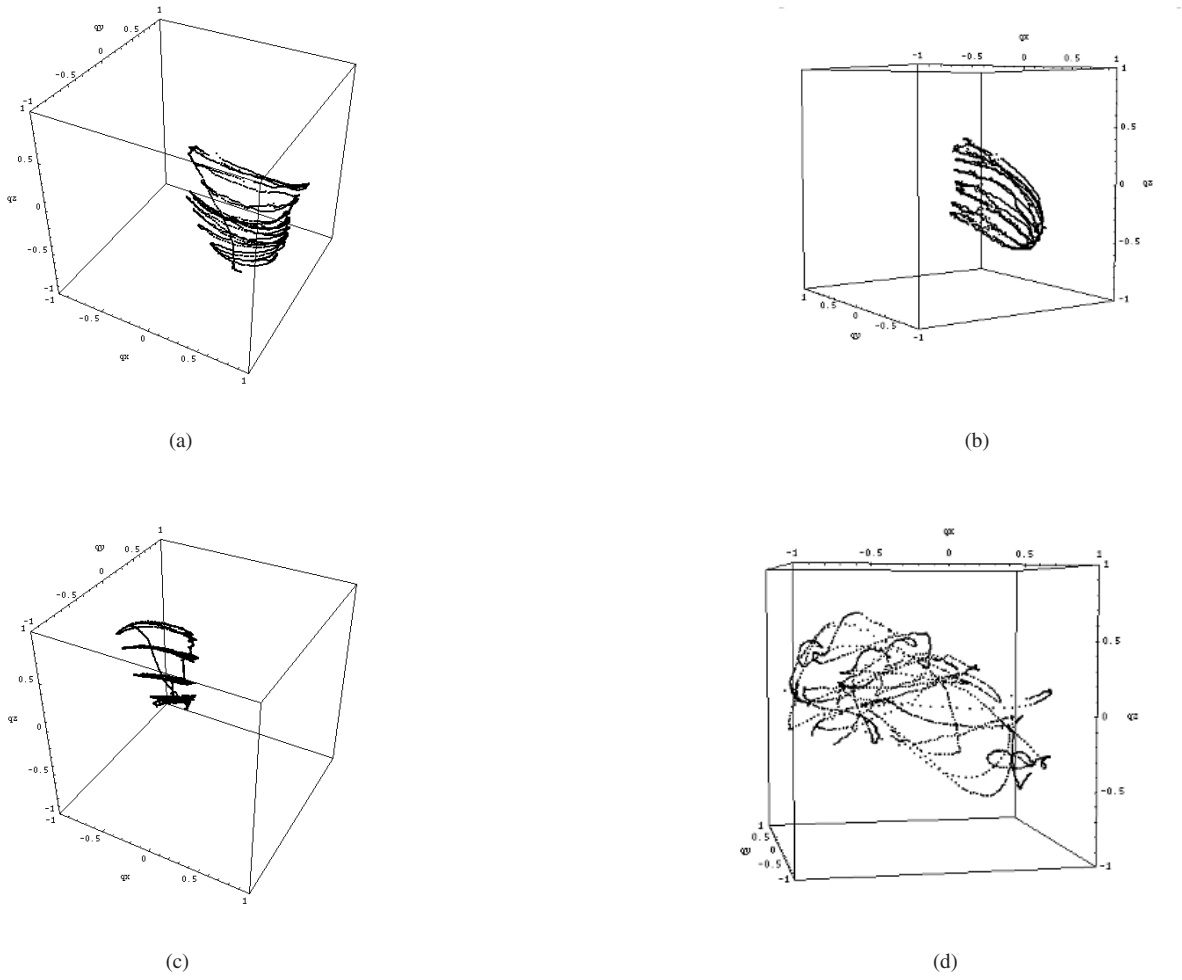
Our reasons for preferring the quaternion rotation formalism over all others in this particular context are:

- 3D rotations are expressed in a very compact form, as they all lie within a 3D sphere in SO(3)[3], the space of three-dimensional rotations, or on the surface of a 4D sphere, in quaternion space;
- rotations in quaternion space are continuous and from this immediately follows that this rotation space provides us with a convenient distance metric.

To further explicit the second statement, what we mean by continuous, is that, up to a scaling factor, the spatial location of discreet quaternions reflects their actual similarity. This is clearly shown in any of the examples of Figure 5.8, where we can readily follow the motion trajectory in 3D space, due to their continuity. Given that the axis-angle rotation formalism is also expressed by a rotation around an arbitrary axis, its 3D representation, the exponential map, presents similar continuity properties but is unfortunately hampered by singularities (see Appendix) and was therefore not considered.

Given this continuity property of quaternions, a distance metric representing a proximity measure is easily defined. In 4D quaternion space, the distance between two quaternions corresponds to the arc-of-circle on the surface of the 4D sphere, and in SO(3), the Euclidean distance between them provides a good approximation of the original geodesic metric [Lawton and Beard2001].

[3]SO stands for "Special", meaning that the equivalent 3×3 rotation matrix R has a determinant of 1.0, and "Orthogonal", meaning that RR^T is equal to the identity matrix.



(a) (b)

(c) (d)

Figure 5.8: Decomposing the motion: (a) flexion/extension; (b) abduction/adduction; (c) twist at four different arm elevations; (d) random upper arm motion.

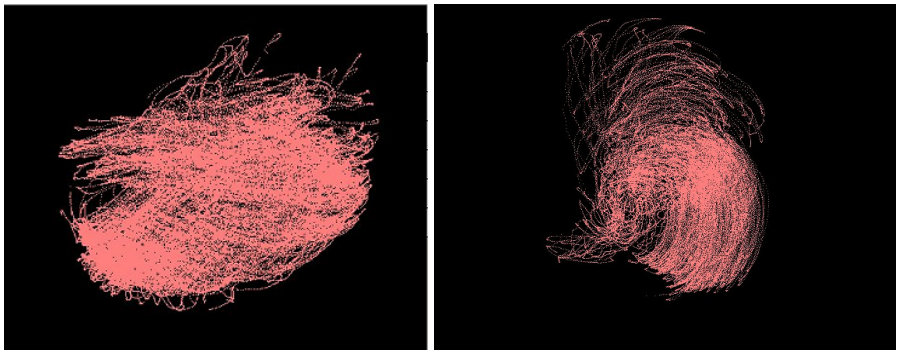
Hence, if we succeed in defining an enclosing surface around the rotations, we will be able to determine the closed set of valid quaternion rotations, and thus have derived a joint limits representation that encloses both the angular and axial rotation components .

As we are herewith presenting the theory behind our joint limits representation, the processing details being revealed in the next chapter, we show, with no further explanation, the cloud of 3D quaternions corresponding to shoulder motion, depicted by Figure 5.9(a) and (b).

Having obtained the volumetric data corresponding to valid quaternion rotations, we now approximate it by an implicit surface, the choice of this surface representation being justified in the next paragraph.

5.3.1.2 Implicit surfaces

The shape reconstruction problem is an issue that has been tackled over and over again, and therefore a great variety of methods can be found. These can be divided into two main categories, depending on whether the chosen surface representation is mesh-based or implicit.



(a) (b)

Figure 5.9: (a) and (b) 3D quaternion data for the shoulder ball-and-socket joint, from two different view points.

In the case of a mesh representation, methods exist for meshes with fixed or non-fixed topology. For reconstructing the shape of scattered 3D data, a new triangulated mesh can be created or an initial mesh deformed to fit the data. An enumeration of such methods, as well as their drawbacks, is given in [Bittar *et al.*1995]. These methods are not convenient for representing our joint limits as the surface has a discreet representation (a list of polyhedra, polygons, triangles, edges, etc.). This would involve lengthy browsing and computation for determining whether a point is inside or outside, or which is the closest point on the surface.

Implicit surfaces, on the other hand, allow to express a shape using relatively few parameters, while at the same time providing a convenient metric for estimating proximity. This is due to the fact that the surface is represented by an equation and not modelled explicitly. This representation has thus gained an increasing appeal in the area of shape reconstruction. As well as presenting a new technique, the work of [Zhao *et al.*2000] also provides an overview of what has been done to this day in terms of reconstruction using implicit surfaces. The recent trend has seen an increase in research in the direction of variational implicit functions in n-dimensional space, such as hyper-surface representations [Gomes and Mojsilovic2002] or radial-basis functions [Carr *et al.*2001, Turk and O'Brien2000]).

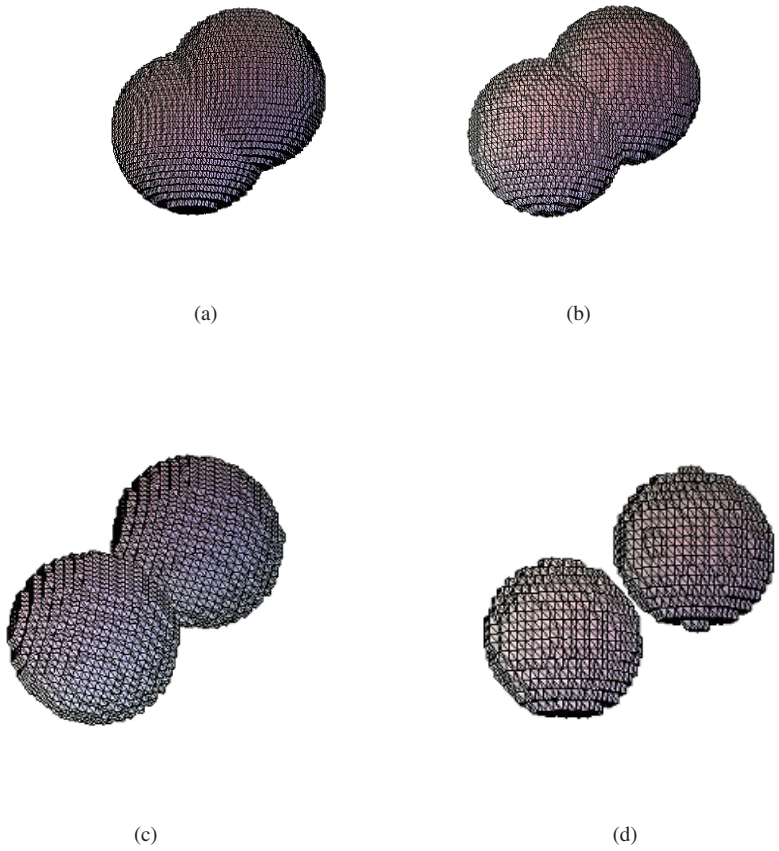
In our case, not only will an implicit surface provide us with a smooth representation of the volume of valid quaternion rotations, but thanks to its well-defined distance function between data points and the obtained surface, the fitting of such a surface to data points is relatively simple, as is the inside/outside decision. Furthermore, when dealing with a value representing an invalid rotation, the use of implicit surfaces gives us the ability to easily project this value to the closest valid one.

Implicit surface parameterisation For the interested reader, a good overview of implicit surfaces can be found in [Opalach and Maddock1995]. To explain the notion in a nutshell, an implicit surface S is a set of primitives surrounded by density fields f_i from which an iso-surface is derived eq.(5.3).

$$S = \{\ P(x,y,z) \in \Re^3 \mid F(P) = iso\ \}$$

$$F(P) = \sum_{i=1}^{n} f_i(P) \tag{5.3}$$

The skeletal primitives can be points, lines, curves or polygons and the overall shape of the primitive follows the shape of the skeleton. The point-based primitive is the most common type used, its density field being built around it. Density functions vary depending on the shape reconstruction problem they apply to, the general categories being:



(a) (b)

(c) (d)

Figure 5.10: Effects of parameters of field functions, for a fixed e=0.5: (a) k=0.5; (b) k=1.0; (c) k=2.0; (d) k=3.0.

- blobbies:

$$f_i(P) = e_i.e^{-k_i r} \tag{5.4}$$

- metaballs:

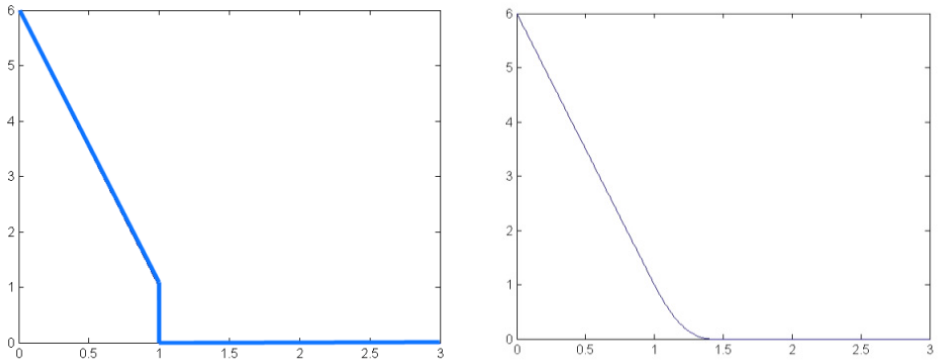
$$f_i(P) = \begin{cases} k_i(1 - \frac{3r^2}{e_i^2}) & \text{if } r \in [0, \frac{e_i}{3}] \\ \frac{3k_i}{2}(1 - \frac{r}{e_i}) & \text{if } r \in [\frac{e_i}{3}, e_i] \\ 0 & \text{if } r \in [e_i, \infty] \end{cases}$$

- soft objects:

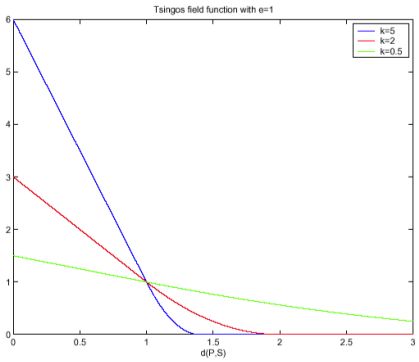
$$f_i(P) = \begin{cases} k_i(1 - \frac{4r^6}{9e_i^6} + \frac{17r^4}{9e_i^4} - \frac{22r^2}{9e_i^2}) & \text{if } r \in [0, e_i] \\ \\ 0 & \text{if } r \in [e_i, \infty] \end{cases}$$

where $r = d(P, S_i)$, and d is the Euclidean distance to the primitive's centre. The f_i field functions are parameterised by:

- the iso-value *iso* of the surface that determines how close the envelope sits with respect to its primitives. The higher the iso-value, the tighter the surface will fit the primitives;



(a) (b)



(c)

Figure 5.11: Density functions with respect to *r* for a fixed e=1.0: (a) Tsingos' density function cut off at the iso-value (1.0), for k=5.0; (b) our modified progressively decreasing field function, with the same iso-value and stiffness; (c) plot of the field function for various values of k.

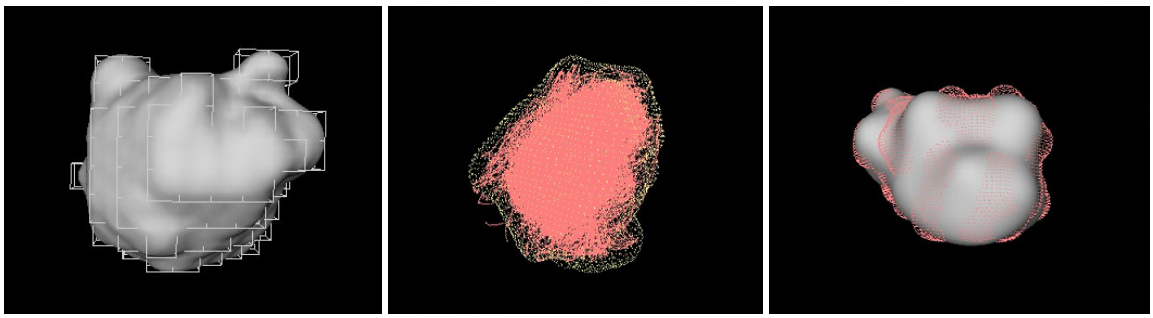
- the radius e_i (or thickness) of the sphere S_i created by a single primitive so that $f_i(e_i) = iso$;
- the stickiness k_i that defines the blending properties of the surface.

For an overview of various possible field functions, see [Wyvill and Wyvill1989]. The effect of varying the stiffness for two blobbies is shown in Figure 5.10, where the field function used is $f_i(P) = e^{-k_i(r-e_i)}$.

Implicit surfaces were used by [Tsingos *et al.*1995] to reconstruct medical organs, and our approach is directly based on their surface definition and reconstruction process. We take the implicit surface S to be an iso-surface, or contour surface, of a field defined by one or more spherical primitives of field function:

$$f_i(P) = \begin{cases} -k_i r + k_i e_i + 1 & \text{if } r \in [0, e_i] \\ \frac{1}{4}\left[k_i(r - e_i) - 2\right]^2 & \text{if } r \in [e_i, R_i] \\ 0 & \text{elsewhere} \end{cases} \tag{5.5}$$

where $R_i = e_i + \frac{2}{k_i}$.



(a) (b) (c)

Figure 5.12: Implicit surface for shoulder compound data: (a) extracted iso-surface with an iso-value of 5.0; (b) the quaternion data and its enveloping implicit surface; (c) simplified implicit surface with a reduced number of primitives.

The expression of the radius of influence R_i has been modified with respect to the expression published in [Bittar *et al.*1995] and [Tsingos *et al.*1995], which initially was $R_i = e_i - \frac{2}{k_i}$. If we plot this field function (Figure 5.11(a)), we see that the function is abruptly cut off at an iso-value of 1.0, whereas our function decreases progressively (Figure 5.11(b)).[4]

The reader who is familiar with such shape reconstruction methods will be wondering how such techniques could be usable for our data, given that our quaternion rotations are volumetric, and shape reconstruction methods apply only to surface data.

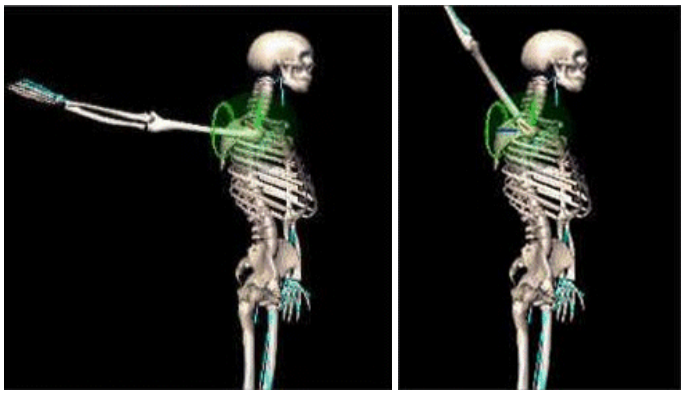
This was indeed a problem that we had to overcome, and the details of how this was done are given in Chapter 6 that deals with all the practical issues of our method. At the present moment, we will consider the process as a black box and just present the implicit surface corresponding to the quaternion data shown in Figure 5.9. The resulting implicit surface envelope is depicted in Figure 5.12, including a simplified version of the implicit surface that can be obtained if the initial number of primitives is deemed too high. How this simplification was carried out is also included in the chapter relating the processing details.

5.3.1.3 Practical use of implicit surface joint limits

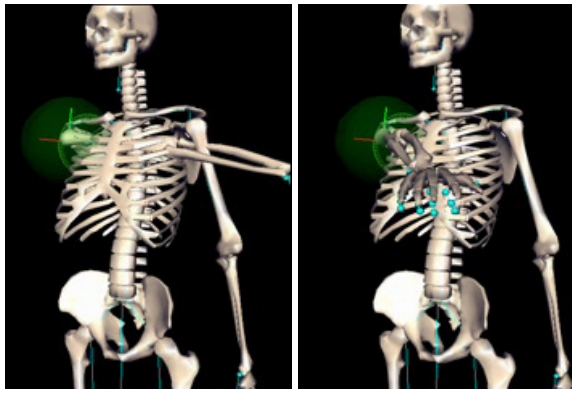
In spite of the fact that the ultimate aim of our joint limits is to prune the state-space of postures in the context of motion capture, their usefulness can also be demonstrated in the case of computer graphics, namely when it comes to animating a virtual human. We will consider both cases and show some results obtained using the joint limits for the shoulder compound shown in Figure 5.12.

Animation To test the joint limits in an animation context, we will simply apply joint rotations to the ball-and-socket shoulder joint, in the direction of the main motion components (abduction/adduction, flexion/extension and axial rotation), showing where the effective limits are reached (Figure 5.13). When the applied rotation exceeds the boundaries of the implicit surface joint limits, a procedure is applied in order to re-project an invalid rotation onto the surface of the boundary envelope. How this is done is explained in the chapter on implementation details.

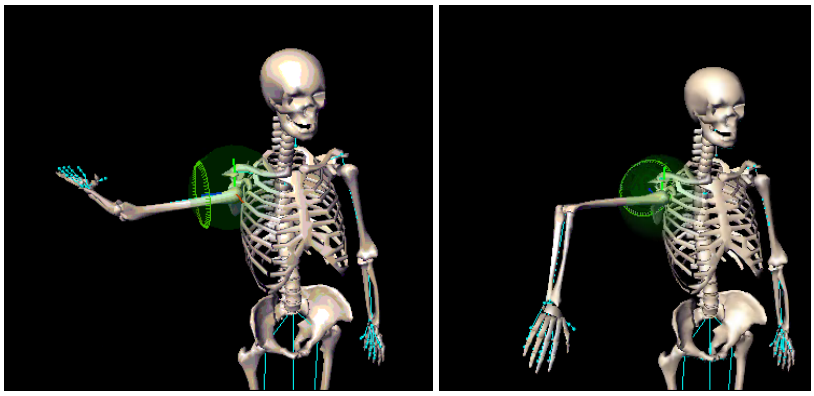
[4] Whether this is intentional or not, we do not know for sure, as we received no answer from the author concerning this matter.



(a) (b)

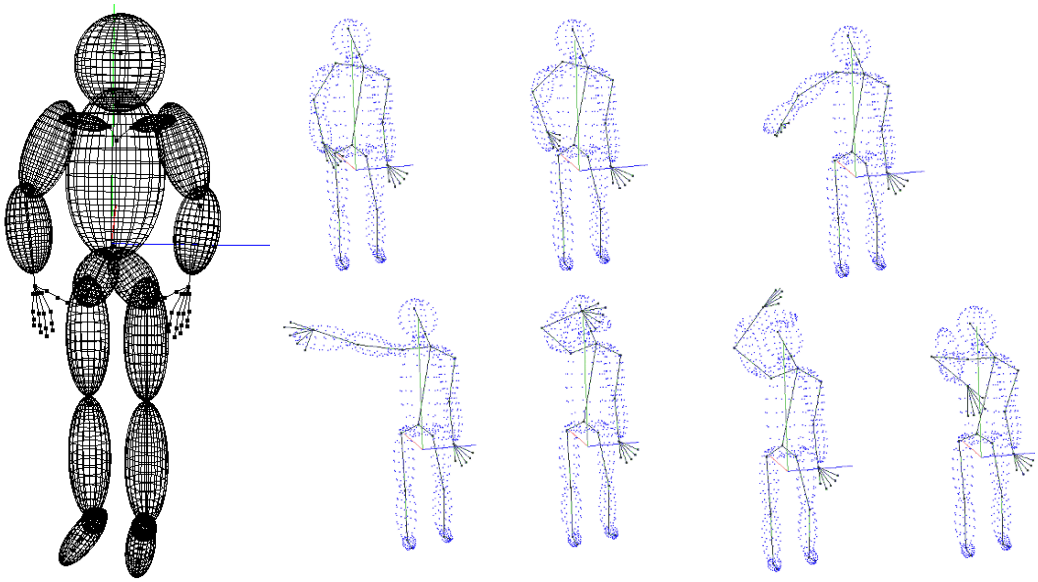


(c) (d)



(e) (f)

Figure 5.13: Applying joint limits during animation: maximal abduction (a) without and (b) with joint limits; flexion (c) without and (d) with; inner axial rotation (e) without and (f) with.



(a) (b)

Figure 5.14: Posture recovery with shoulder compound joint limits: (a) metaball-fleshed out model used; (b) posture output, superposed with synthetic data.

Posture recovery from synthetic 3D data For carrying out this experience, we designed a keyframe sequence, generating synthetic data on the basis of the shape primitives used to flesh out the body parts of our model. These shape primitives are metaballs, whose field function is defined as per eq.(5.4), such a body model having been successfully applied to stereo data motion capture by [Plänkers and Fua2001].

In Figure 5.14, we show the result of the fitting, the model being displayed without shape primitives in order to make the rotations at the joint level more clearly visible. We compare the results obtained with or without applying any joint limits, and note that sporadic shoulder rotations have disappeared in the latter case. Random rotations still appear at the elbow level, as no joint limit enforcement has been carried out on that joint, this problem being remedied using coupled joint limits, as per the following sub-section.

5.3.2 Joint limits for coupled joints

We underlined in Section 5.1 the importance of being able to express joint coupling in a joint limits representation. The implicit surface formalism illustrated in the case of the single ball-and-socket shoulder joint can readily be extended to reflect such a joint inter-dependency.

We will prove this statement by applying the method to the coupled joints that are present in the human upper limb, namely the sterno-clavicular and gleno-humeral joint coupling on the one hand, and the gleno-humeral and elbow joint coupling on the other. Once the joint limits for both couples have been determined, we will have obtained a fully constrained model of the upper limb joints that contribute to motion in such a way that their impact is visible in terms of surface deformation.

The two afore-mentioned joint sets have the following configuration in terms of dimensionality:

- the sterno-clavicular / gleno-humeral joint couple is composed of a 2D ellipsoidal joint followed by a 3D ball-and-socket joint;

- the gleno-humeral / elbow joint couple is composed of a 3D ball-and-socket joint followed by a 2D pivot joint.

As we mentioned before, the implicit surface joint limits representation can just as well be applied to a 2D case as to a 3D case, a 2D implicit surface having the same properties as its 3D counterpart. The only difference resides in the fact that the centres of the primitives are defined in 2D instead of 3D, and that the resulting iso-surface is a contour instead of a surface.

If we wish to capture joint inter-dependency in terms of rotations, we need to measure the range of motion of both joints simultaneously, and express the rotations of the child joint with respect to its parent in the articulated structure hierarchy. If we consider the two separate quaternion clouds obtained, the relationship that links the two is the knowledge that each quaternion of the child joint corresponds to a very specific quaternion of the parent joint.

One can of course approximate each cloud separately with an implicit surface, in which case we would obtain the joint limits for each joint, but they would no longer reflect the existing inter-dependency. In order to counter this, we have come up with two suggestions, whose implementation and practical application will be explored in turn. The proposals are:

1. to consider the link that exists between each parent joint quaternion and child joint quaternion as a single 6D item, and deal with the joint limits issue in this dimension;

2. to design a hierarchical representation relating the parent joint limits to child joint limits.

5.3.2.1 Joint limits in higher dimensions

The procedure we have used to approximate quaternion rotations by an implicit surface is generic in terms of dimensionality, meaning that if we consider the collected rotations of a series of joints as an n-dimensional data set, we can approximate it by an n-dimensional implicit surface without any further modifications.

Once the measured rotations for two coupled joints are converted to quaternions, we can consider the vectorial components of the quaternions to form a 6-dimensional vector representing a configuration for this joint set. However, the problem this poses regards the distance metric. Indeed, the Euclidean distance is not extendible beyond three dimensions. Furthermore, the definition of an implicit surface is itself based on this Euclidean metric. One might naively think that one could use the sum of the Euclidean distances at respectively, the parent joint and the child joint levels.

Such thinking leads us to Murphy's Law applied to Generalised Euclidean Distances: "if it seems too good to be true, then it probably is". If we attempt to re-formulate the distance metric in this manner, and define a 6D implicit surface based on the metric, testing such joint limits in an animation context at first seemed to work quite well. However, at some point we happened upon the expected caveat, namely that this metric is meaningless in 6D. The simple proof of this was that the spherical primitives composing the 6D implicit surface were not always included in the surface, in terms of the distance metric. The conclusion that we draw from this experience is that implicit surfaces, as they are presently defined and used in computer graphics, are limited to a three-dimensional space, and cannot be generalised. As to the problem of deriving a distance metric in n-dimensional space, it is a current hot research topic and in general requires dimensionality reduction.

However, our n-dimensional representation is not altogether useless, in the sense that it is based on an n-dimensional voxelisation, which could be applied to defining the space of valid vectors, such a representation readily allowing to determine whether a point is included or not within the set of voxels.

A 6D voxelisation representation would not be usable for animation purposes, as an invalid rotation needs to be projected to the closest valid one, thus once again requiring a metric.

In terms of posture recovery from 3D data, however, it would be applicable as an *a posteriori* constraint on rotations. By this we mean that the parameter set computed at each optimisation step can be tested for validity with respect to the 6D voxelisation, and if the test fails, a large error can be returned, thus dissuading the optimisation of pursuing its search in this direction.

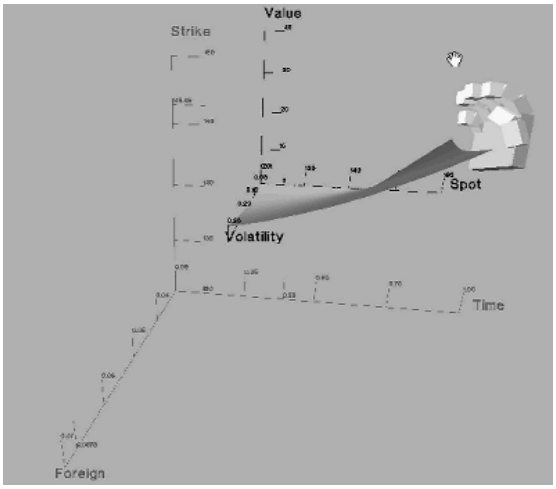


Figure 5.15: Call option whose value is defined as a height field in a 3D inner world, with parameters "spot price" and "volatility". The containing world has three parameters, namely "maturity", "strike price" and "foreign interest rate". All in all, five out of the six parameters are represented in this manner [Feiner and Beshers1990].

With respect to our initial objective, which was to derive an implicit expression for joint limits, it would be the understatement of the year to say that this solution is not very satisfactory, and we will therefore present our alternative proposal, namely hierarchical joint limits.

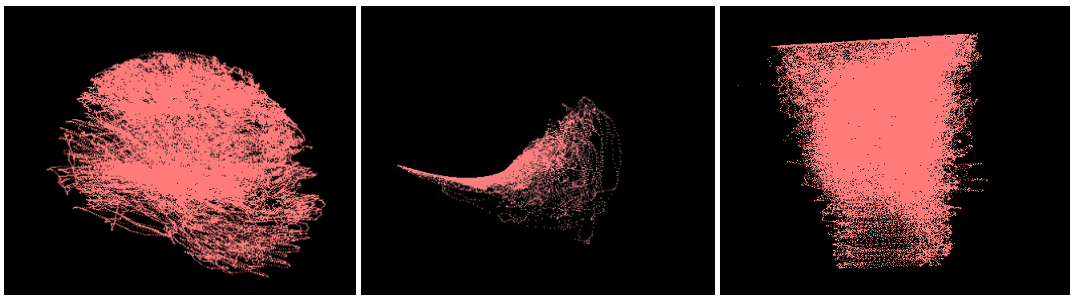
5.3.2.2 Hierarchical joint limits

We have designed a hierarchical joint limits scheme based on the voxelisation of the space covered by the quaternions of the parent joint, each voxel so defining a local cluster of similar parent joint positions (keyframe voxels). For each such cluster, we then compute the corresponding implicit surface of the child joint rotations (keyframe surface), this sub-set of rotations being directly derived from the relationship existing between parent and joint quaternion rotations. If we wish to refine this representation so as to ensure a more smooth transition between child joint limits from one parent voxel to another, we compute intermediate child joint limits using morphing ("morphological metamorphosis") between keyframe surfaces.

The idea was inspired by the notion of worlds-within-worlds, or nested worlds, used in financial visualisation by [Feiner and Beshers1990]. In their test-bed case, the value of an option is function of six parameters, and one would like to know for which set of parameters the value is maximal. Many methods for reducing the complexity of n-dimensional data do so by keeping one or two of the variables constant. The best-known example probably is projection pursuit [Cook *et al.*1997], where the data is projected into two or three dimensions. Feiner and Beshers propose to not perform such a projection, but to manipulate the data directly in 3D by nesting worlds within other worlds. For this, they create a 3D world, representing a set of three parameters, in which they embed another world. The origin of the latter with respect to the containing world's co-ordinate system specifies the value of three of the inner world's parameters that were held constant while sub-dividing the n-dimensional world.

In the end, we have multiple copies of different worlds within a containing world, where each copy, based on its position, has a different constant set of values of the containing world's parameters. Each parameter can be manipulated at will, allowing the visualisation of the change induced at the global level. For the option example in their paper, a 3D world within a 3D world embedding is represented by Figure 5.15.

In a similar way, if we want to express the joint limits of two successive joints, we have a problem that exceeds the third dimension, but that we would nevertheless like to reduce to sub-sets of 3D problems. Therefore, for a succession of joints, we suggest to nest the joint limits of each within those of its parent joint. We will from here



(a) (b) (c)

Figure 5.16: 3D quaternion data from the coupled motion capture of the gleno-humeral and elbow joints: (a) gleno-humeral joint data; (b) elbow joint data; (c) elbow Euler angle data.

on consider the joints two-by-two, but it should be understood that the method can be extended to a succession of as many joints as one likes, the bottleneck being the amount of data available, as at a certain depth, it is bound to become too sparse for implicit surface approximation to be possible.

The 3D-parent-2D-child joint couple We start out with the example of the gleno-humeral and elbow joint coupling as the parent joint is ball-and-socket, thus making the transition of the general implicit surface method to 3D-2D joint coupling more direct.

Joint coupling at this level is not only due to anatomical reasons, but also to the physical presence of the rest of the body, namely the thorax and the head. Indeed, at certain positions of the gleno-humeral joint, the amount of flexion is limited due to the fact that the arm cannot penetrate the thorax and the head. This is typically the case when the upper arm is fully flexed, and as the reader can verify for him/herself, flexion at the elbow level is limited to at most $-\frac{\pi}{2}$ radians (or $-90\,\mathrm{deg}$). At the anatomical level, the amount of twist available at the elbow is linked to the current upper arm elevation, i.e. to shoulder abduction and/or flexion, as was shown in [Wang *et al.*1998].

The gleno-humeral and elbow joint rotations were measured simultaneously, the latter being expressed in the local co-ordinate system of the gleno-humeral joint. The resulting quaternion clouds are shown in Figure 5.16(a)(b).

As can be seen from these images, the elbow joint quaternions are predictably locally flat, due to the fact that motion is carried out around two axes only. Given that the two motion components are independent one from another, and that the flattened quaternion shape will not be easily approximated by a 3D implicit surface, we transform the elbow quaternions into Euler angles. As we are not dealing with a 3 DOF rotation, this conversion is acceptable as the mathematical flaw inherent to Euler angles will not hamper us in the 2D case. The resulting cloud of data is dense and the conversion from its equivalent quaternion uniquely defined within the $[-\pi, \pi]$ interval. Using the Euclidean distance as a metric does in this case not cause discontinuity problems, and will thus be used to represent the distance between 2 DOF Euler angle rotations.

The conversion from quaternions yields a 2D cloud of data, which we then approximate by a 2D implicit surface. The corresponding Euler angles are depicted in Figure 5.16(c).

To reflect the coupling that exists between the two joints, the following *modus operandi* is defined:

- compute the 3D implicit surface defining the joint limits of the parent joint,
- sub-divide the space of valid parent joint rotations into voxels (3D grid),
- for each such voxel, collect the data for the child joint and approximate this by a 2D implicit surface,

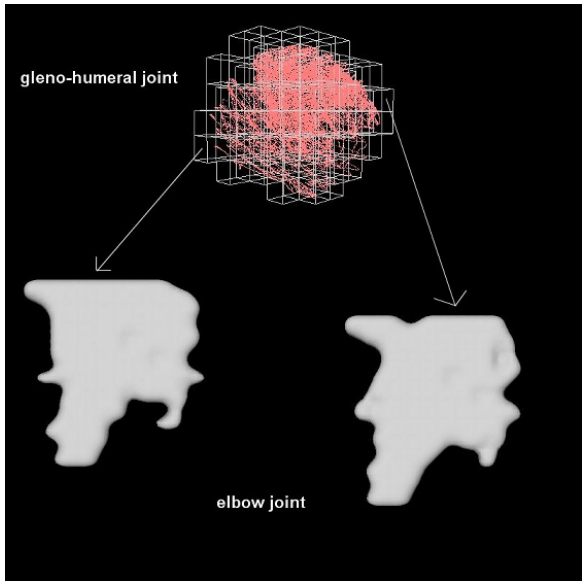
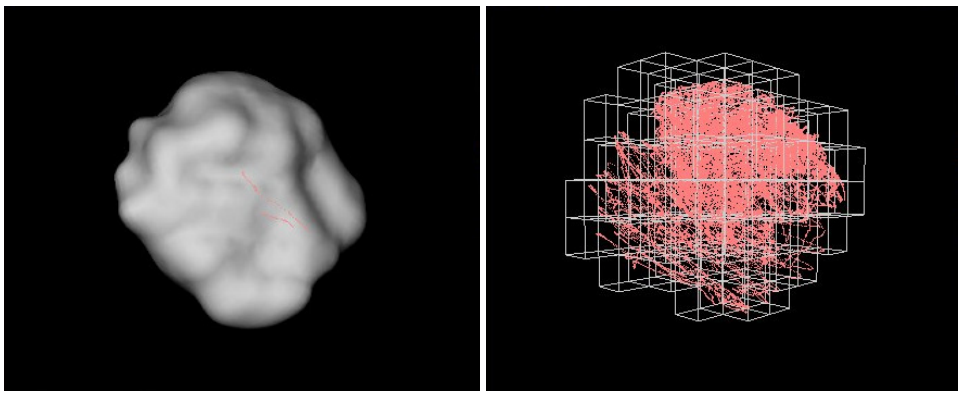


Figure 5.17: Hierarchical joint limits.



(a) (b)

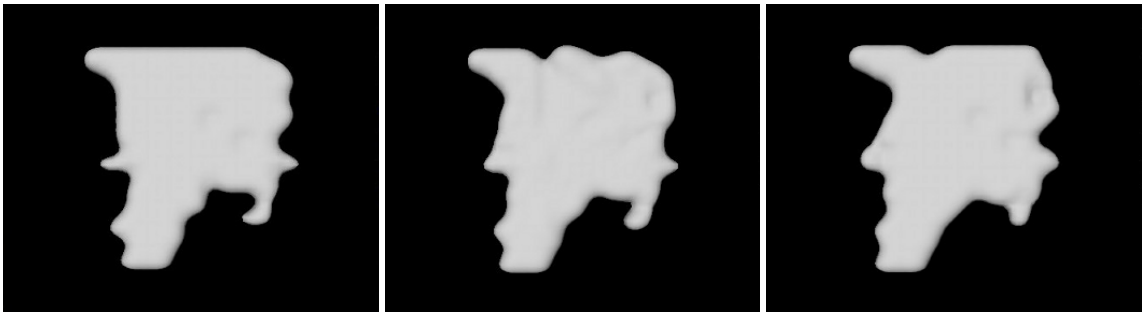
Figure 5.18: Joint limits for the gleno-humeral joint: (a) extracted implicit surface; (b) voxelisation of the gleno-humeral joint quaternions.

- produce intermediate parent joint voxels and corresponding child joint surfaces using morphing,

the entire concept being illustrated by Figure 5.17.

For the gleno-humeral joint quaternion data shown in Figure 5.16(a), we obtain the implicit surface shown in Figure 5.18(a), and the voxelisation for a resolution of $7 \times 7 \times 7$ as in Figure 5.18(b).

For each voxel for the gleno-humeral joint, we obtain a set of Euler angle rotations for the elbow joint, this data set resembling in shape the cloud shown in Figure 5.16(c). Variations occurring at the boundaries of the gleno-humeral joint voxelisation, where the range of motion of the elbow joint is reduced, are either due to joint anatomy or to the limiting effect of other body parts. Each such set is approximated by a 2D implicit surface, using exactly the same technique as was applied to the 3D case. As joint limits should vary smoothly from one voxel to another, as they do in reality, we refine the voxelisation of the gleno-humeral joint data by



(a) (b) (c)

Figure 5.19: Interpolation of two elbow joint limits: (a) source implicit surface, located in voxel *i*; (b) shape obtained half way through the morphing process; (c) target implicit surface, located in voxel *i+1*.

inserting additional voxels between each two voxels of the gleno-humeral joint voxelisation. We compute the corresponding elbow joint implicit surfaces by morphing the two elbow implicit surfaces that were associated to the initial two gleno-humeral voxels. In Figure 5.19, we show an example of a source and target elbow joint implicit surface and the intermediate surface computed using morphing.

The 2D-parent-3D-child joint couple Having derived joint limits for the first set of coupled joints in the human upper limb, we will now deal with the second set, the sterno-clavicular and gleno-humeral joints. This case of a 2D rotational joint coupled with a 3 DOF joint presents no further complications with respect to the configuration of the previous joint set, and will be handled in the following fashion:

- the quaternions of the sterno-clavicular joint will be converted to Euler angles and approximated by a 2D implicit surface;
- the Euler angles will be voxelised;
- for each voxel, the 3D implicit surface envelope will be defined for the corresponding set of gleno-humeral joint quaternions;
- the 3D gleno-humeral joint implicit surfaces will be morphed in order to increase the resolution of the joint limits.

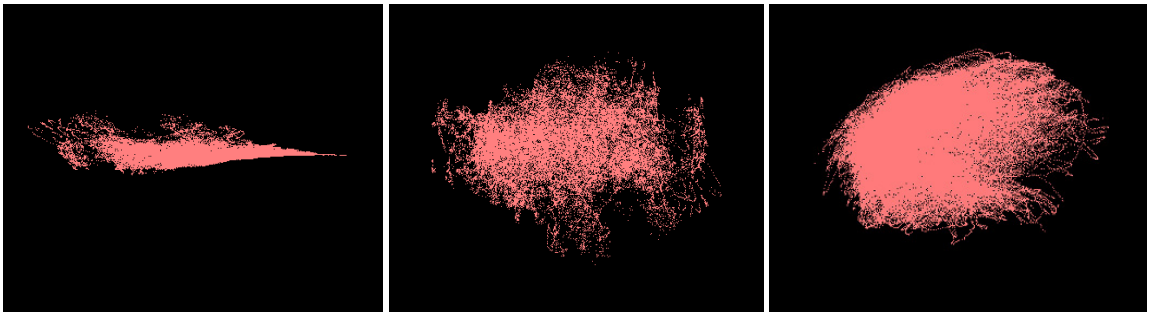
The results are presented in the same order as for the previous joint set.

Figure 5.20 shows the computed quaternions for the sterno-clavicular joint and the corresponding Euler angle data set, as well as the associated gleno-humeral joint quaternions.

The Euler angle data is approximated by a 2D implicit surface, and a 2D 10×10 voxelisation of the data is carried out to determine the sterno-clavicular joint keyframe voxels, this being depicted by Figures 5.21(a) and (b). The final morphing step is applied to the gleno-humeral joint implicit surfaces, defined for each voxel of its parent joint (Figure 5.22).

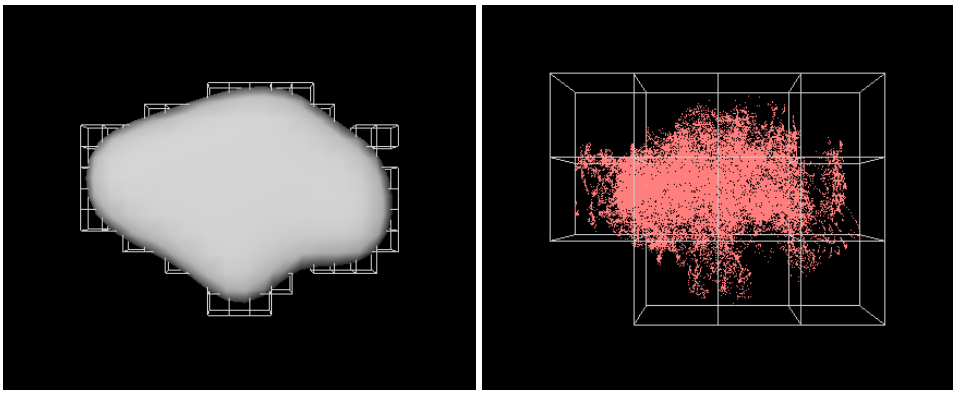
5.4 Enforcing hierarchical joint limits

In the previous sub-section, we derived joint limits for two sets of coupled joint in the human upper limb, and will now enforce these constraints in various types of applications. Joint limit constraints can be applied either *a posteriori* or *a priori*. In the first case, the joint limits serve the purpose of preventing the joint rotations



(a) (b) (c)

Figure 5.20: 3D quaternion data from the coupled motion capture of the sterno-clavicular and gleno-humeral joints: (a) sterno-clavicular joint data; (b) corresponding Euler angle data; (c) gleno-humeral joint data.



(a) (b)

Figure 5.21: Sterno-clavicular joint data: (a) 2D implicit surface corresponding to the Euler data; (b) voxelisation of the sterno-clavicular joint Euler angles.

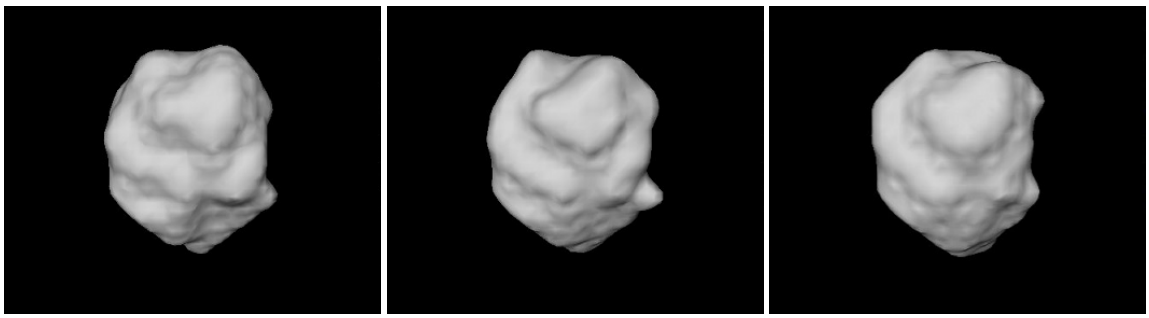
from taking on values outside the defined boundary, which is typically the case in video-based motion capture. *A priori* constraints are used for character animation, where any invalid rotation is immediately clamped to its closest valid solution on the surface of the joint limits boundary.

5.4.1 Hierarchical gleno-humeral and elbow joint limits

5.4.1.1 Character animation

The hierarchical joint limits for this coupled joint set are applied as *a posteriori* constraints in the following manner:

- for a given gleno-humeral rotation, determine whether it is contained within the 3D implicit surface boundary defined for this joint;
- correct the posture, if it turns out to be outside the surface;



(a) (b) (c)

Figure 5.22: Interpolation of two gleno-humeral joint limits: (a) source surface; (b) intermediate surface; (c) target surface.

- for the elbow joint, determine in which voxel its parent (gleno-humeral) joint is currently situated and verify the elbow joint rotation with respect to the child joint implicit surface corresponding to the voxel;
- correct the rotation, if it is outside the surface.

In the context of character animation, we demonstrate the usefulness of our joint limits in two ways:

1. by moving along the main motion components of the two inter-dependent joints to test the effective constraining when the limits are reached, for each joint.
2. by applying the joint limits to a keyframe sequence where motion boundaries have voluntarily not been respected.

For the first test, we used the animation application that will be further described in Section 7.4 concerning implementation details. The hierarchical joint limits for the gleno-humeral and elbow joints are loaded, and the two joints are moved as per the figures on the following pages.

Figure 5.24: in the six frames shown in this Figure, we only test gleno-humeral motion, and wish to visualise the limits in each direction and their subsequent clamped values, respectively inner twist, outer twist, flexion, extension, abduction and adduction. As one can see, when reaching the boundaries of inner or outer twist, the clamped position is not a pure outer twist value, but also contains a partial abduction or adduction component. This can readily be explained by the fact that what mathematically corresponds to the nearest valid solution does not correspond to what we would consider to be the most intuitive closest solution. Indeed, when trying to reach maximal twist, we are moving along one of the axes of the co-ordinate system, and if we move beyond the boundary of the implicit surface, we would logically expect the clamped solution to be the point where the axis intersects the implicit surface. This is however not the case given that in terms of Euclidean distance, this is not the closest point. The drawing in Figure 5.23 should make this clear.

Figure 5.25: in the next four frames, the gleno-humeral joint is set at its initial pose, and the elbow joint is moved in the direction of maximal flexion, extension, inner twist and outer twist. We can see that maximal extension is zero, as the elbow joint has no extension range, and we note again the presence of twist and flexion composition when clamping maximal inner twist.

Figure 5.26: in this frame, we modify the gleno-humeral joint position. As can be seen in the messages displayed in the console window at the bottom right, the current voxel number of the gleno-humeral joint is updated as the joint is rotated. The corresponding joint limits are then loaded for the elbow joint.

Figure 5.27: in the last five frames, we explore the joint limits of the elbow joint for the given gleno-humeral joint position. In the first frame, we push the flexion component to its maximum, and we note that the range of

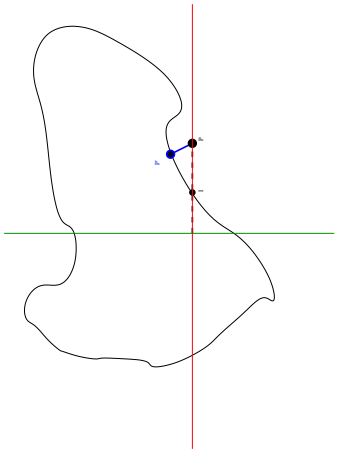


Figure 5.23: Explanation of the apparent drifting of the clamped position when moving along one of the co-ordinate axes: in this 2D representation of an implicit surface, we are moving along the vertical axis, we reach the point P outside the boundary. The intuitive clamped position would be I' at the intersection between the axis and the implicit surface. However, the mathematical solution is P', and given that its 2D position has two non-zero values, the resulting rotation will also be a composition of two motion components.

motion is not the same as for the previous gleno-humeral position of Figure 5.25. In this case, this is not due to the fact that the joint is limited in its range of motion, but because at some flexion point, the hand reached the thorax, as is shown in the second frame when viewed from the top. In the third frame, we test maximum extension and again note that there is none, and finally we apply maximum inner and outer twist, the motion composition phenomenon being once again present in the case of inner twist.

For our test on keyframe sequences, we exploited the capacity of the animation application to import VRML files for a standard H-Anim compatible humanoid. Key-frame sequences were created in 3D Studio Max and exported to VRML using a plug-in developed at Miralab, University of Geneva. Existing keyframes were modified in order to exaggerate certain motions and to obtain joint rotations that were clearly outside the boundaries.

Figures 5.28 and 5.29: a tennis serve motion, where the left column displays the original out of boundary motion, and the right column, the corrected motion using hierarchical gleno-humeral and elbow joint limits.

Figures 5.30: another motion with exaggerated elbow flexion.

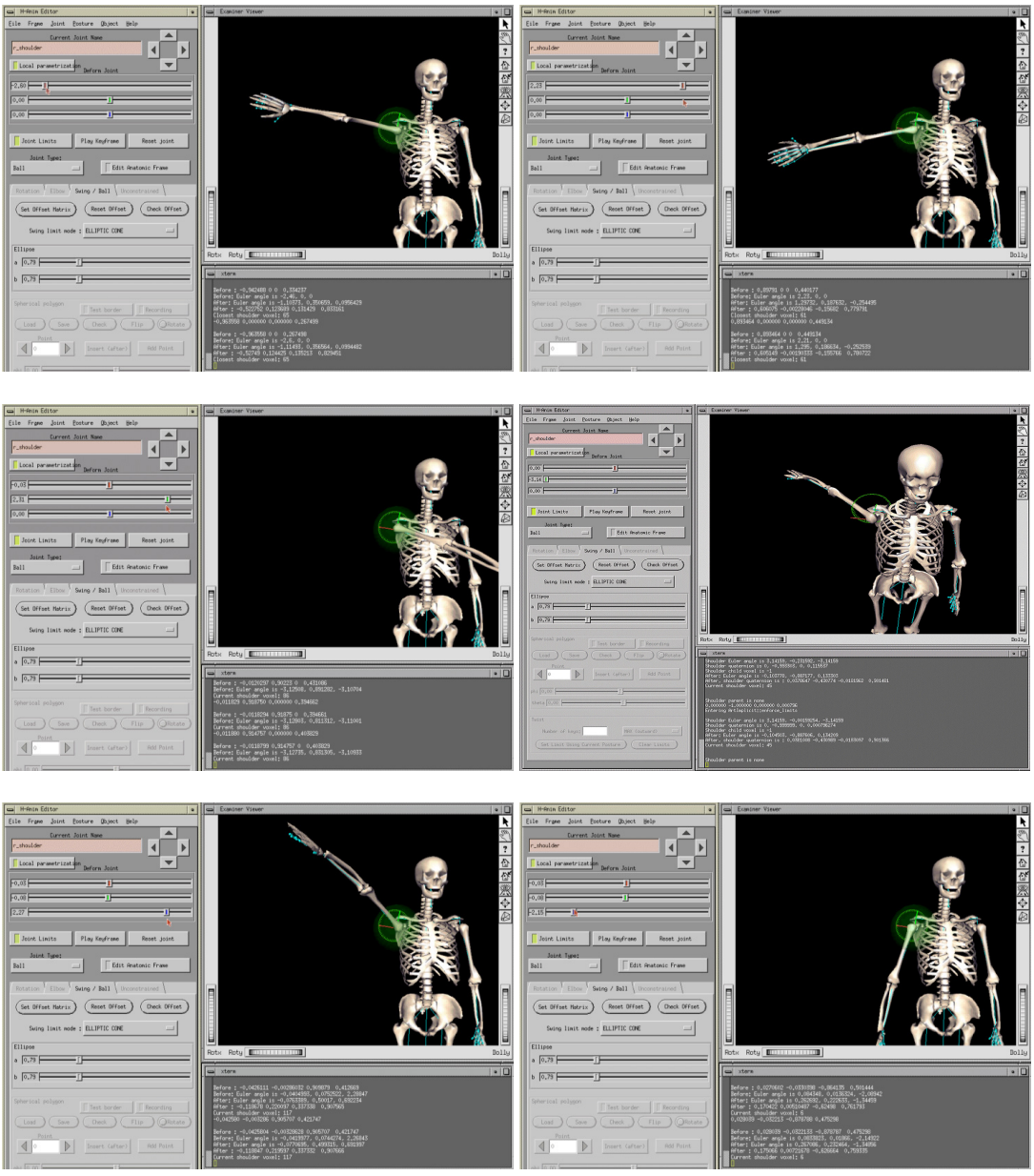


Figure 5.24: Gleno-humeral motion.

Figure 5.25: Elbow motion for initial gleno-humeral position.

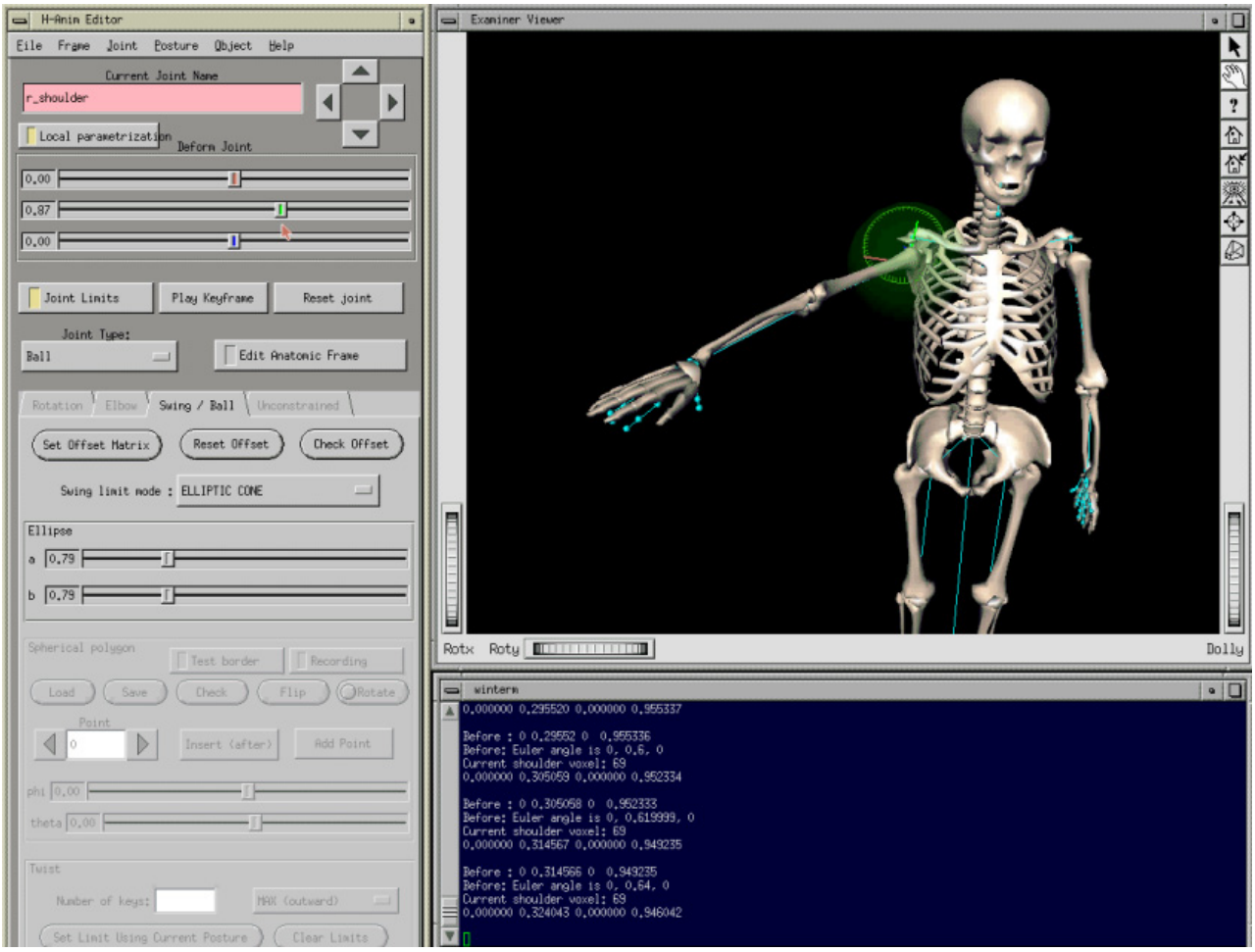


Figure 5.26: Modified gleno-humeral position.

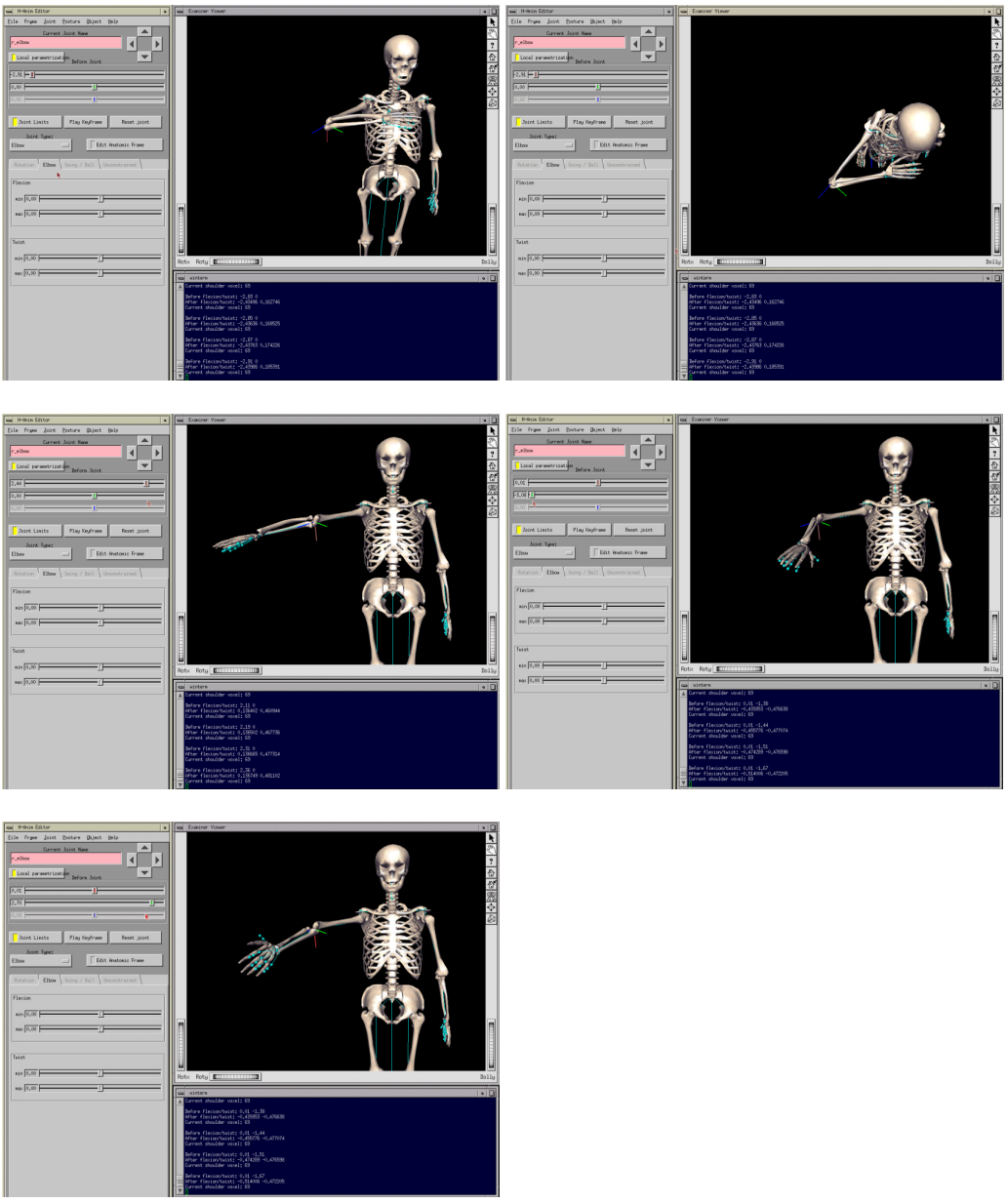


Figure 5.27: Elbow motion and new gleno-humeral position.

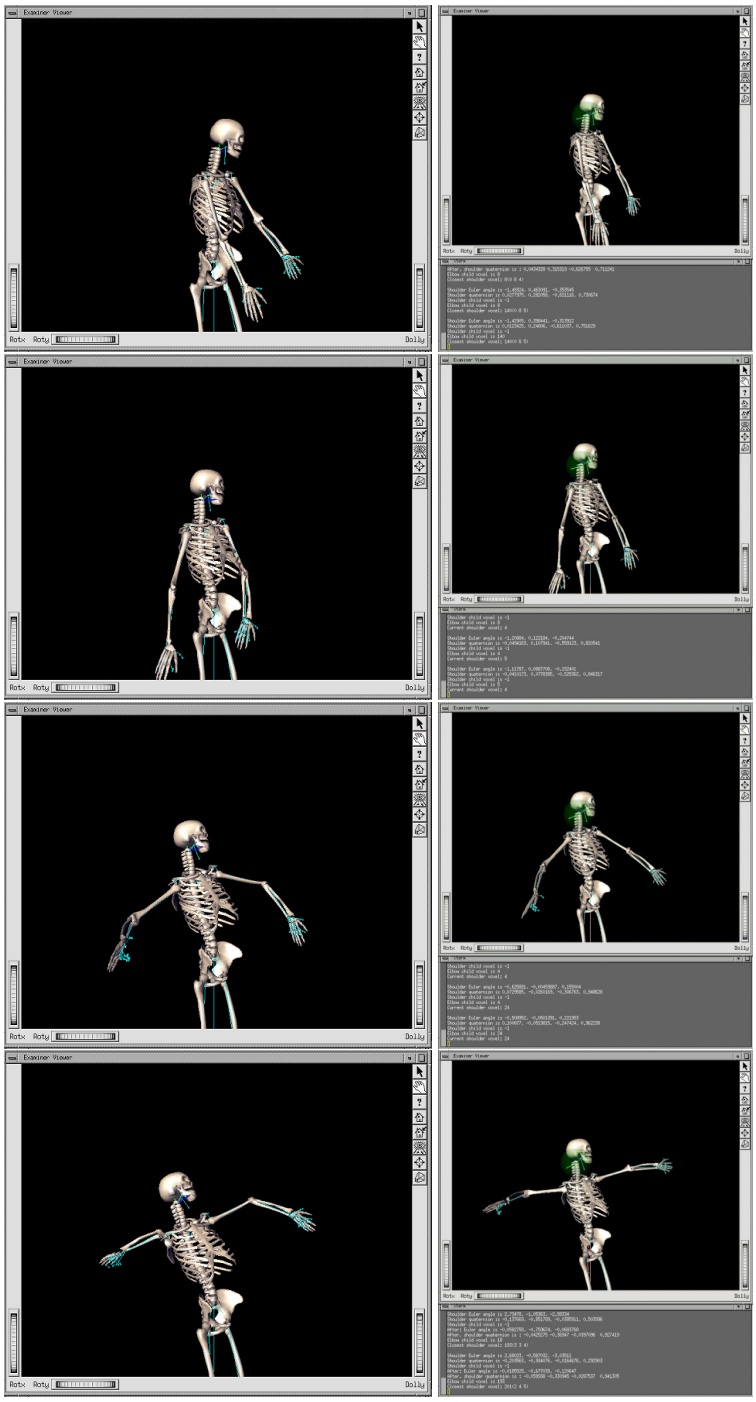


Figure 5.28: Frames of the tennis serve with gleno-humeral and elbow joint limits (part 1).

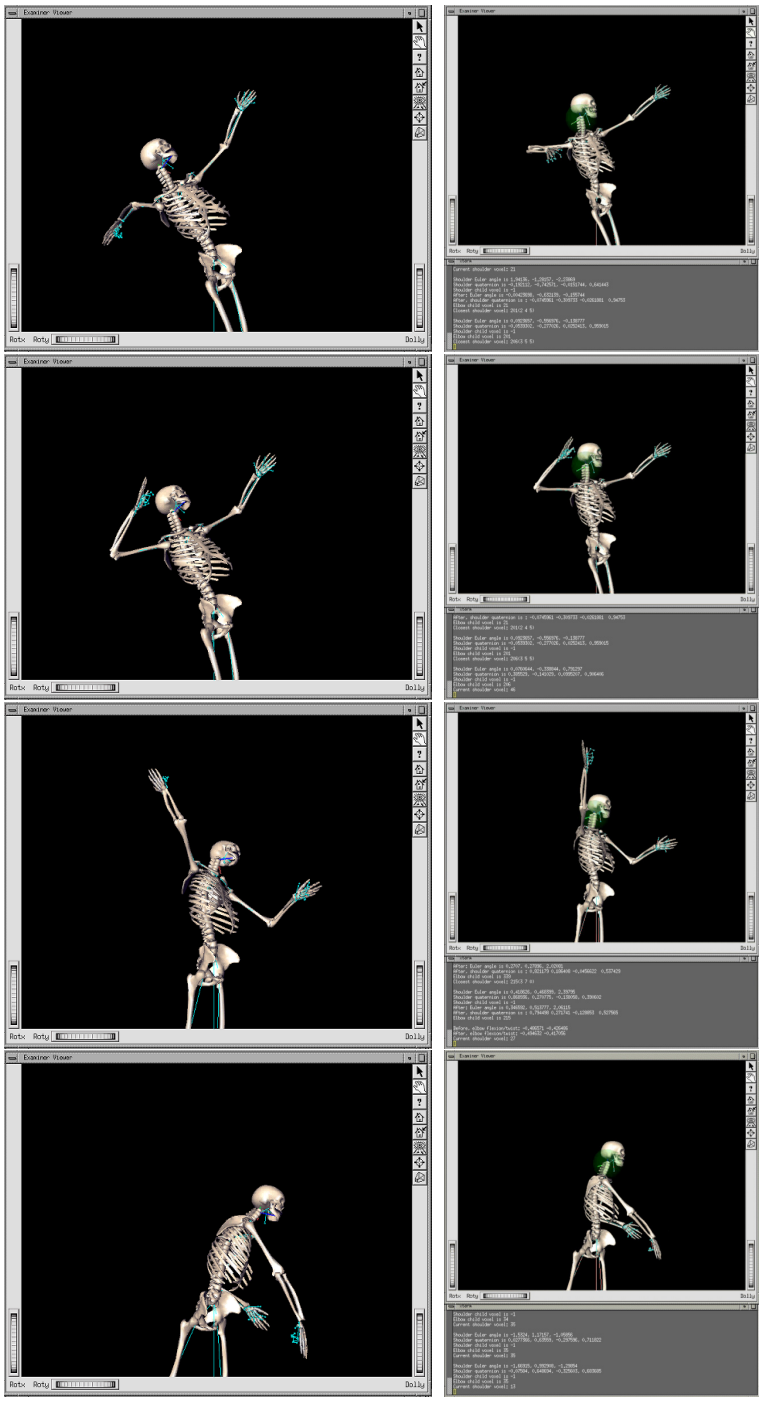


Figure 5.29: Frames of the tennis serve with gleno-humeral and elbow joint limits (part 2).

Figure 5.30: Motion frames “Are you crazy?” with gleno-humeral and elbow joint limits.

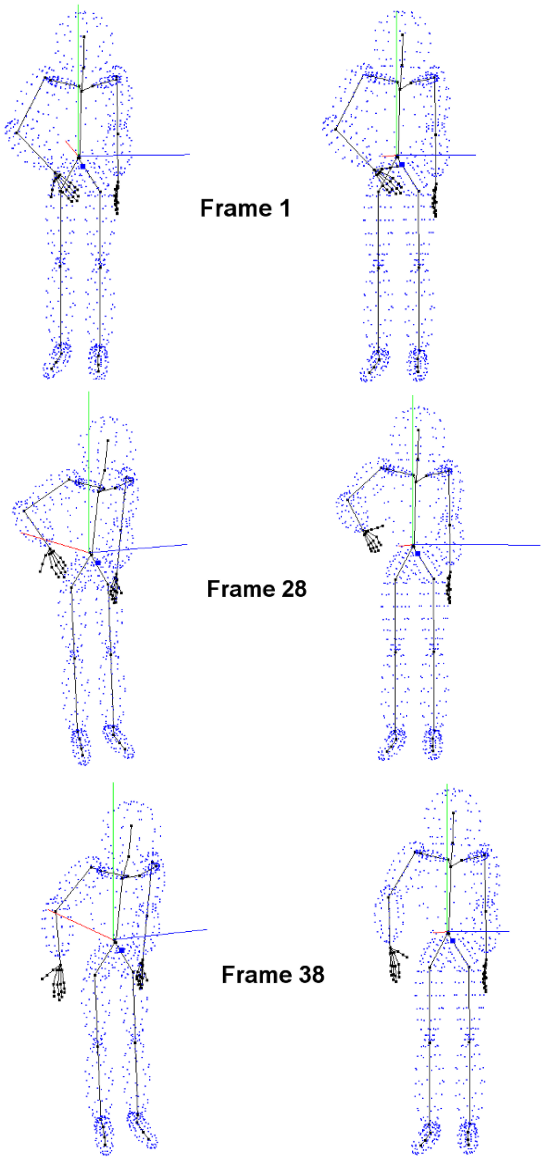
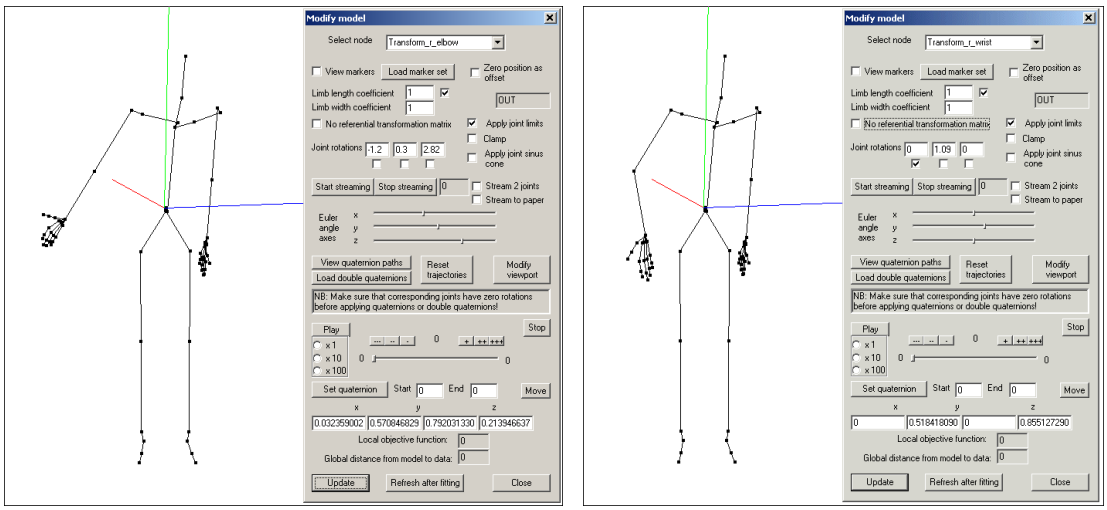


Figure 5.31: Least-squares optimisation for posture recovery on range data. In each figure, the left and right images respectively represent the recovered posture without and with joint limits enforcement. As one can see, the arm is rotated in a very unnatural manner in the absence of joint limits.

5.4.1.2 Motion capture

We created a synthetic keyframe sequence using our H-Anim compatible body model, thereby certifying that the data is perfectly clean. We carried least-squares optimisation on the sequence, first without enforcing any joint limits, and then again using the hierarchical joint limits for the gleno-humeral and elbow joints.

Despite the absence of noise in the data, errors occur over approximately one fourth of the frames. Once an incorrect posture has been computed by the optimiser, without joint limits, it will naturally remain within a close range of the chosen values, unless there is a major change in the data. For the first 50 frames of our sequence of about 200 frames, the parameters at the gleno-humeral and elbow level were both wrong, in terms of joint



(a) (b)

Figure 5.32: Applying the parameters to verify their incorrectness: (a) at the shoulder level, the dialogue box tells us that the posture is outside the joint limits ("OUT" message displayed), the main reason for this being the extreme axial rotation; (b) in order for the least-squares fitting to find an optimal solution, the rotation parameters at the elbow level have to compensate for the twisting at the shoulder level, this corresponding to an extension of 1.09 radians at the elbow. Unfortunately, the elbow joint has hardly any freedom of motion in the extension direction, so here too, we are outside the joint limits.

limits. We illustrate this using three frames within these 50 first, in Figure 5.31. The postures recovered using the joint limits are perfectly correct and the output animation is identical to the original keyframe sequence used to generate the range data.

We have verified that the posture in the images on the left are indeed invalid with respect to joint limits by loading a body model and its joint limits, and interactively applying the corresponding parameters. Our application being able to apply limits in the animation context, we can read directly in a dialogue box whether a given posture is valid or invalid, and if we wish, we may even correct the posture using the limits. In Figure 5.32, we show the result of this analysis, for the case of frame 38, where the gleno-humeral parameters were -1.2 adduction, 0.3 extension and 2.82 outer axial rotation, all expressed in radians. For the elbow, an extension of 1.09 was computed.

We then tested our joint limits on real data extracted using stereo vision on three cameras mounted on a triangular rig. The data we thus obtain is noisy and due to the proximity of the cameras, we lack depth information and obtain only front-side data, as is shown in the images of Figure 5.33.

Due to camera geometry, we had to pay attention to limit the amount of occlusion as this would lead to a total absence of data, typically in the case where a limb is perpendicular to the camera plane. We applied unconstrained and constrained tracking to four sequences. For each motion, we superpose the original video frames with the recovered posture, as well as the skeleton of the body model with the 3D stereo data. The purpose of the latter is to demonstrate that if discrepancies were to be noticed with respect to the posture re-projection onto the video sequence, this is not necessarily due to a bad fit. Indeed, due to occlusion or to a limb being perpendicular to the camera plane, stereo data may be unavailable or very sparse. In the case where even constrained tracking produces occasional sub-optimal results, we probably find ourselves in a singularity position of the constrained least-squares. For further explanations on this issue, see Section 7.3.2.

The skeleton model superposed with the stereo data allows us to verify that the least-squares matching is as accurate as possible, in the current conditions, even if the recovered posture does not appear to be 100% correct.

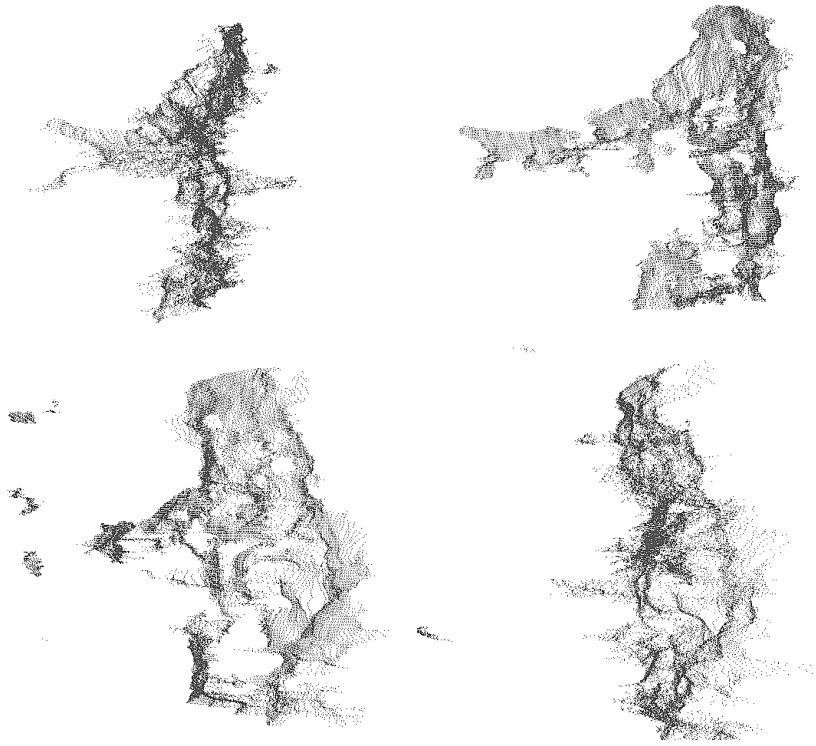


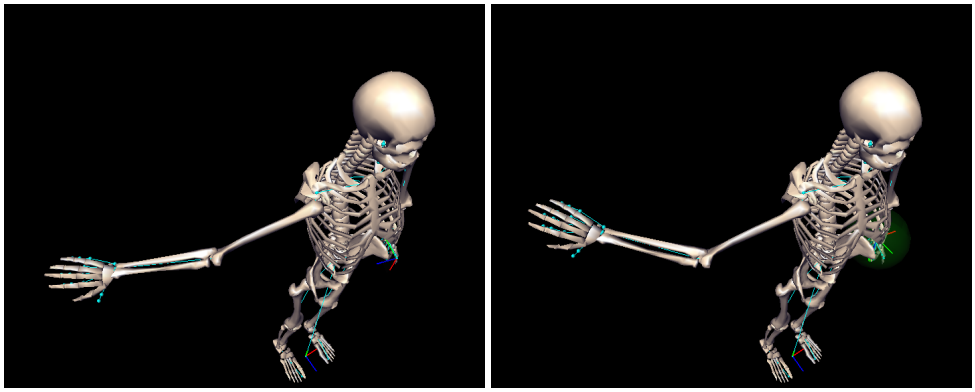
Figure 5.33: Extracted stereo data from a person standing in the capture volume, from various viewpoints.

All sequences have an approximate length of about 100 frames, captured at a rate of 14 frames per second.

Some frames of the first sequence, tracked without any constraints, are displayed in Figure 5.38. Up to halfway through the sequence, until roughly frame 40, tracking is performed correctly. After that point, however, one notices that where an axial gleno-humeral joint rotation and an elbow flexion would have been the correct posture to infer, an elbow extension without shoulder twisting was computed instead. Elbow extension being impossible, this is an invalid posture. Tracking with joint limits enforcement clearly shows this, where for the same frames, the correct posture was found, as shown on the opposite page in Figure 5.39. In this case, the observed tracking error is due to data sparsity. As the arm moves away from the camera, stereo data for the forearm becomes scarce and a precise match is difficult to compute. Towards the end of the sequence, as the arm moves forwards again, data is once again available, but the recovered posture is incorrect, due to the existence of multiple solutions for the same data. A closer look at the tracking errors is shown in Figure 5.34, where the erroneous postures are applied to an anatomical skeleton model.

The second captured motion, similar to the previous one, is shown in Figure 5.40. Up to frame 39, the tracking algorithm handles posture recovery correctly, even without joint limits. However, at frame 42, an elbow extension occurs, which is not allowed by the joint limits and things degenerate from there on. Sporadic axial rotations become present at the gleno-humeral joint level, due to the fact that any amount of twisting equally explains the observations, as shown in Figure 5.35.

In the sequence shown in Figures 5.42 and 5.44, we observe a similar phenomenon, at frame 50, with excessive gleno-humeral joint axial rotation, followed from frame 51 onwards, by invalid elbow joint extension on top of the incorrect shoulder twisting. The errors starting at frame 50 are due to data sparsity, the problem becoming especially acute in the last frame shown, where the quasi-total absence of data leads to an erroneous segmentation of 3D data. As can be seen, the forearm of the model was subject to an attempt to be matched to the data corresponding to the upper arm. The fact that the inferred posture was physically impossible caused this



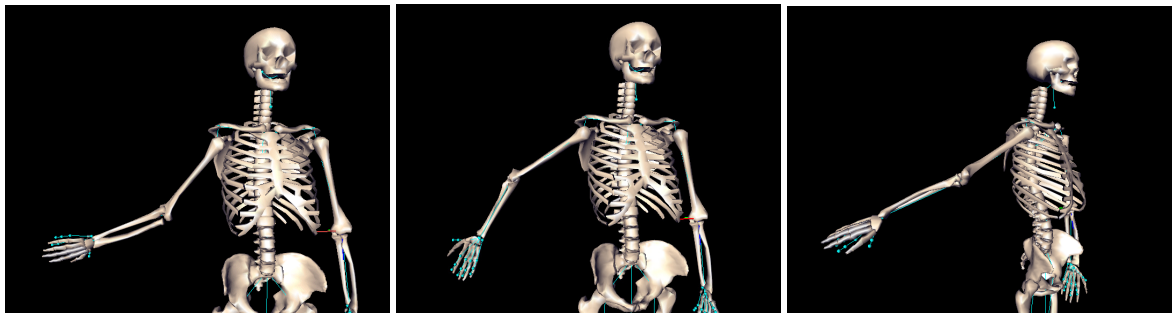
(a) (b)

Figure 5.34: Postures corresponding to frames of the tracked sequence 1 without joint limits: (a) frame 43, with an invalid elbow extension; (b) frame 47 with an even greater elbow extension.

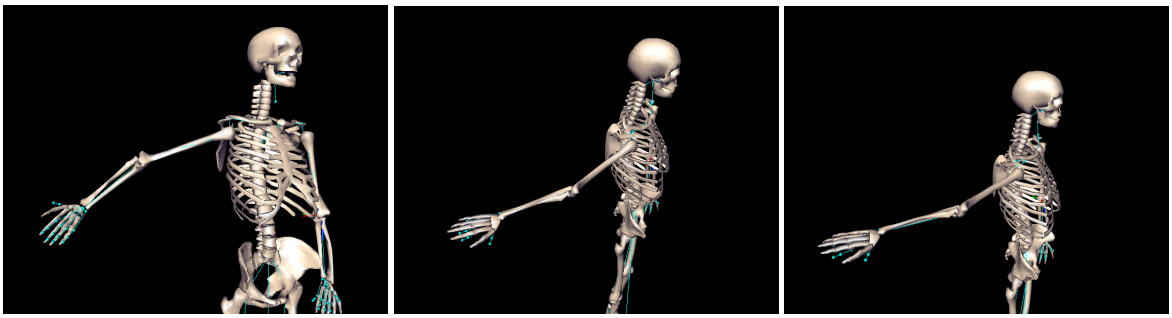
problem to disappear upon joint limits enforcement. The tracking errors are more clearly displayed in Figure 5.36.

A last sequence is shown in Figure 5.46. In this sequence, at frame 1 already, an incorrect axial rotation at the gleno-humeral joint level has been recovered. This error persists and is seconded by an erroneous attempt to match the forearm to the hip 3D stereo data instead of to the forearm data, as in frames 23 to 25. The tracking process recovers from the data association error around frame 31 just to once again compute a posture with an invalid extension at the elbow level. Close-ups on the invalid postures are displayed in Figure 5.37.

As is shown by these tracked sequences, joint limits allow the recovering from various types of tracking errors, which are either due to the lack of data, the erroneous association of 3D data to a body part, or simply multiple state-space solutions for the same data.

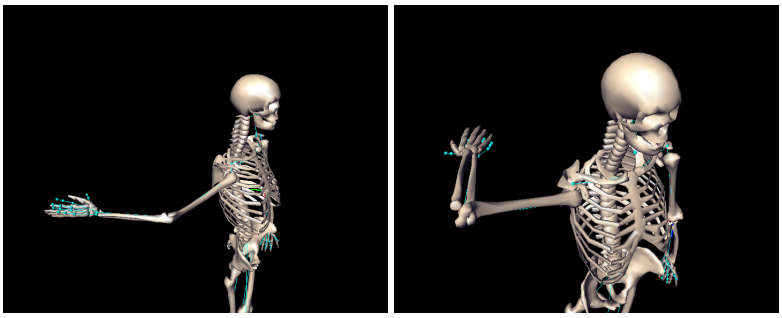


(a) (b) (c)



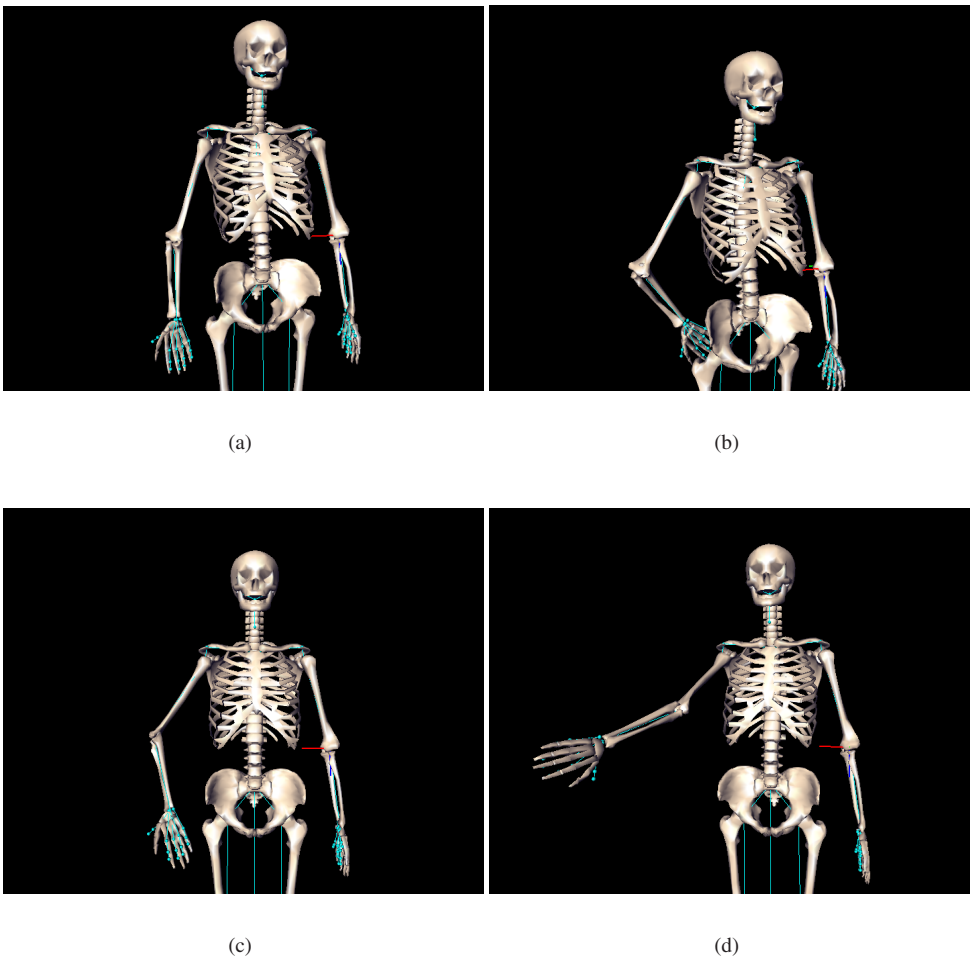
(d) (e) (f)

Figure 5.35: Postures corresponding to frames of the tracked sequence 2 without joint limits: (a) frame 42, with an invalid inner axial rotation of the shoulder, on top of an incorrect elbow extension; (b) frame 43 with the same problems as in the previous frame, except that shoulder axial rotation is extreme in the outer direction; (c) frame 44, shoulder twisting is re-set to a correct value, but invalid elbow extension remains, as well as in (d) frame 45; (e) frame 46 and (f) frame 48.



(a) (b)

Figure 5.36: Postures corresponding to frames of the tracked sequence 3 without joint limits: (a) frame 50, with an invalid outer axial rotation of the shoulder; (b) frame 51, with correct shoulder twisting, but this time an invalid elbow extension.



(a) (b)

(c) (d)

Figure 5.37: Postures corresponding to frames of the tracked sequence 4 without joint limits: (a) frame 1, where a shoulder outer rotation coupled with an invalid elbow extension was inferred, where a simple elbow flexion would have done the trick; (b) frame 23 presenting the same symptoms as frame 1; (c) frame 25, the problem has still not been solved, and furthermore, the outer axial rotation of the shoulder is now totally out of range; (d) frame 34, an invalid inner axial rotation of the shoulder was recovered for this frame, along with an incorrect elbow extension.

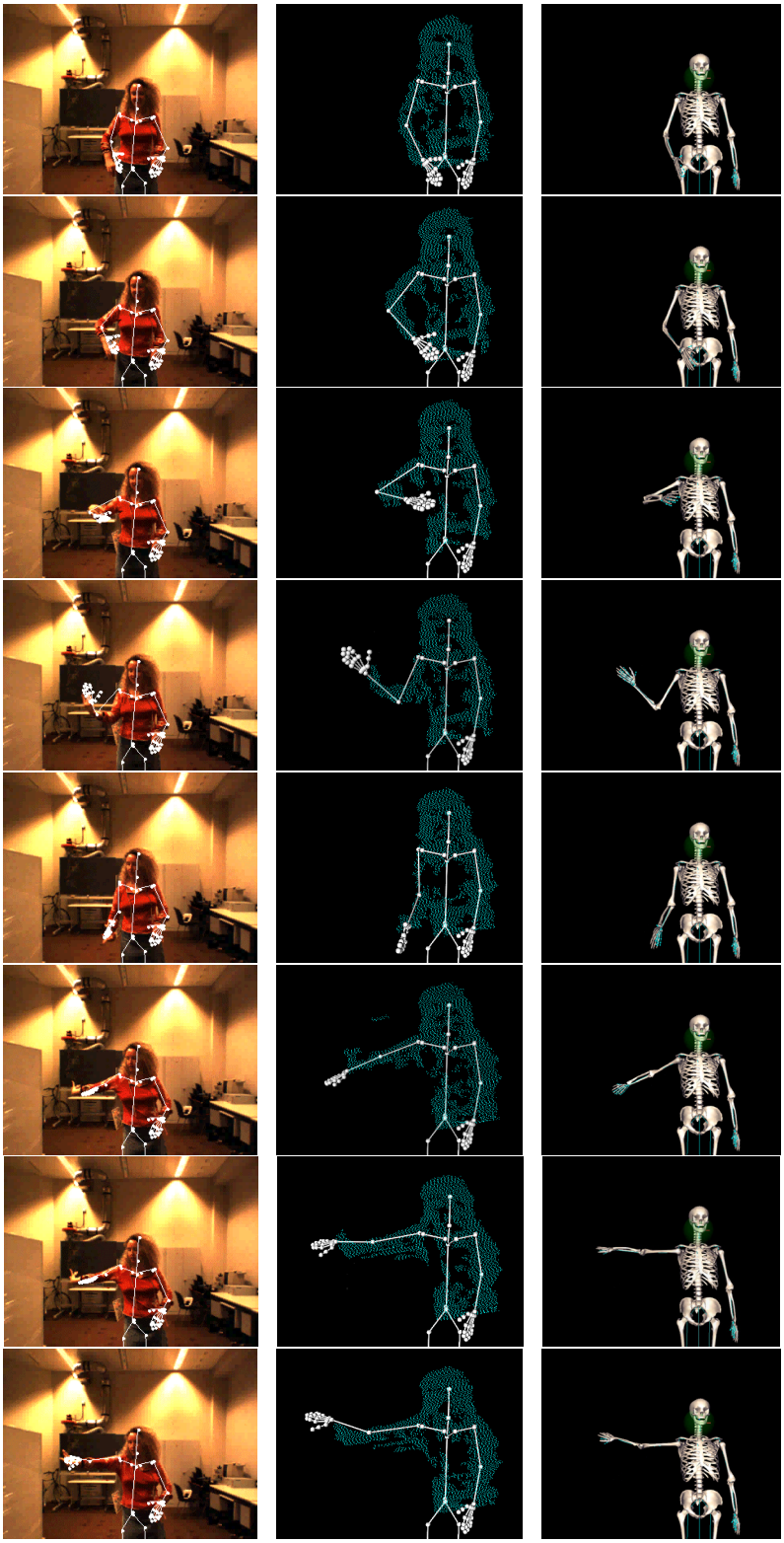


Figure 5.38: Unconstrained tracking of sequence 1. The first column shows the skeleton projected onto the original video sequence, the second column, the skeleton of the body model superposed with the 3D stereo data. The last column displays the recovered posture applied to an anatomical skeleton model. The featured frames are numbered 1, 15, 22, 26, 32, 40, 43 and 47.

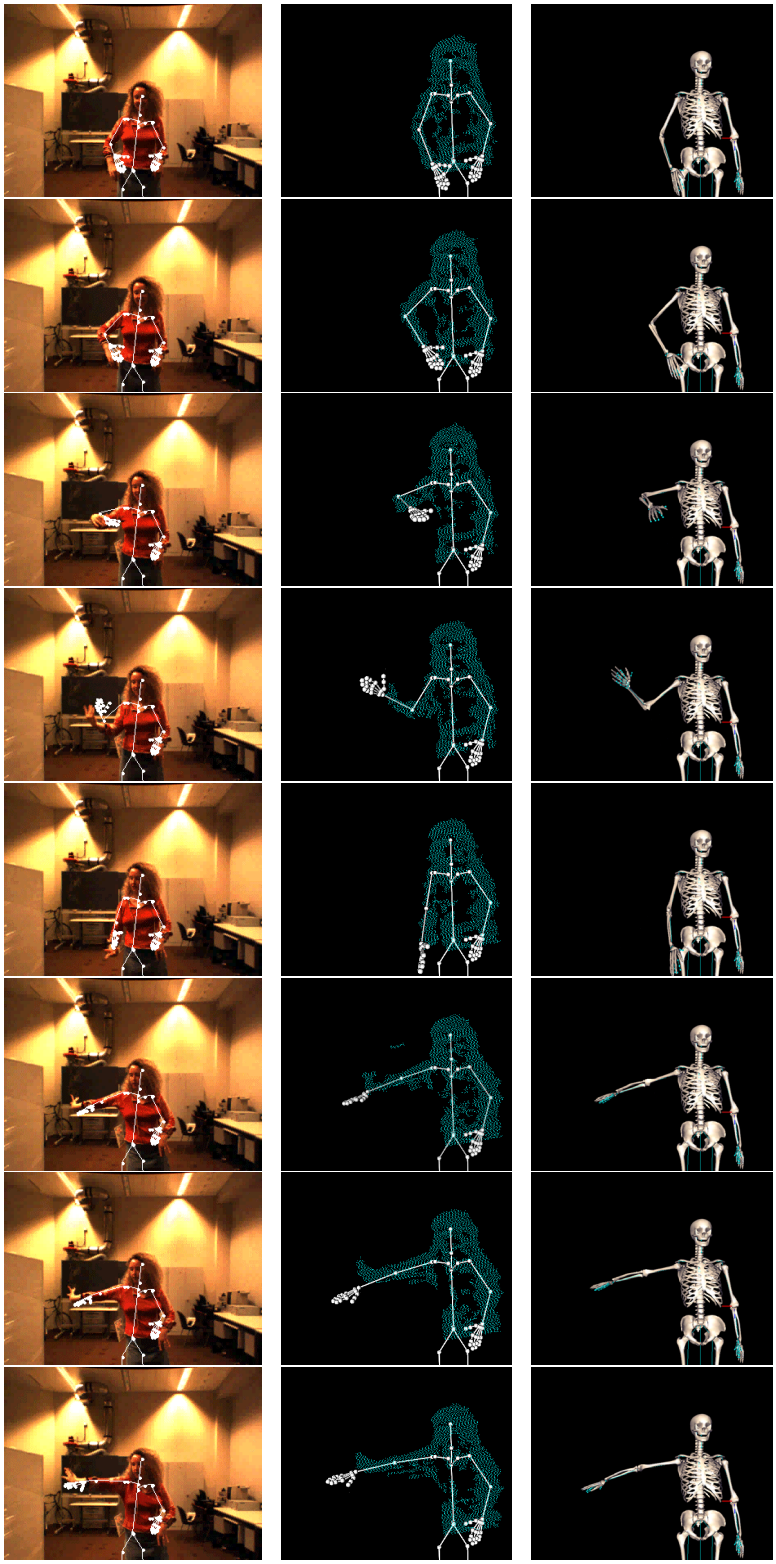


Figure 5.39: Constrained tracking of sequence 1.

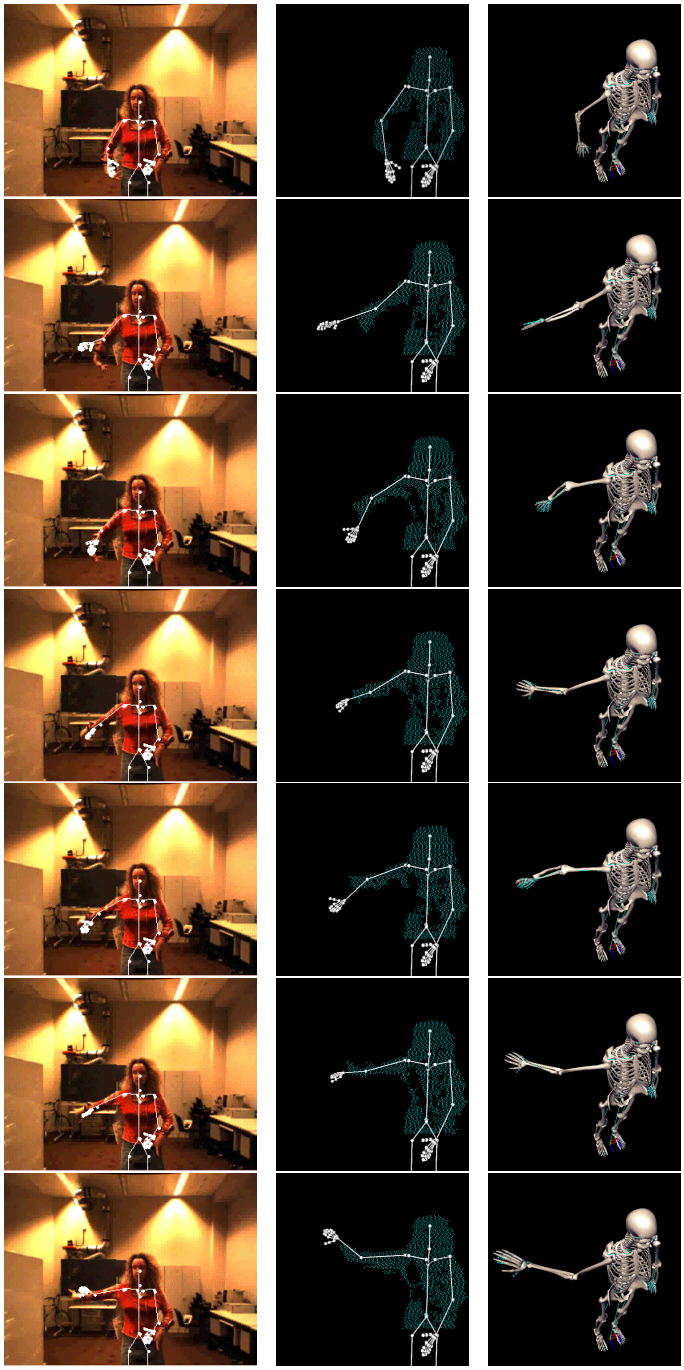


Figure 5.40: Unconstrained tracking of sequence 2. The featured frames are numbered 39, 42, 43, 44, 45, 46 and 48.

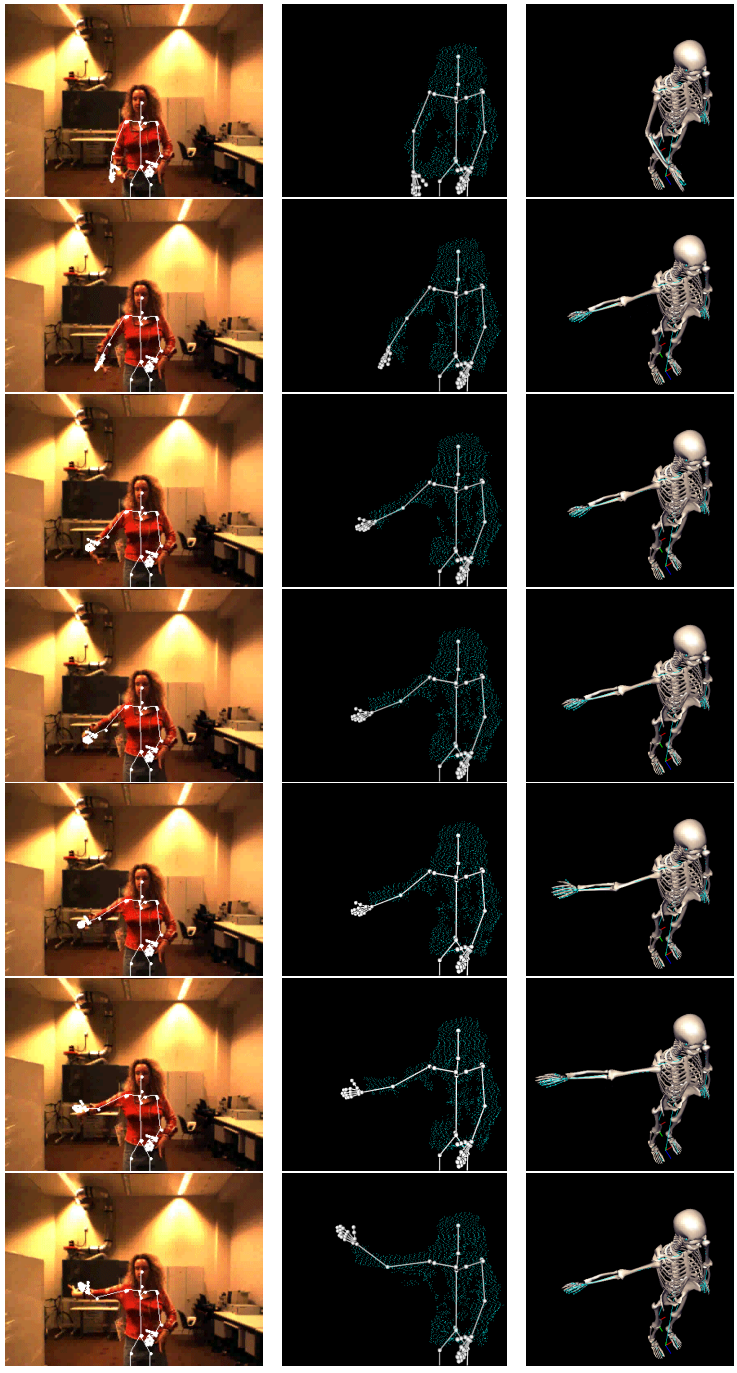


Figure 5.41: Constrained tracking of sequence 2.

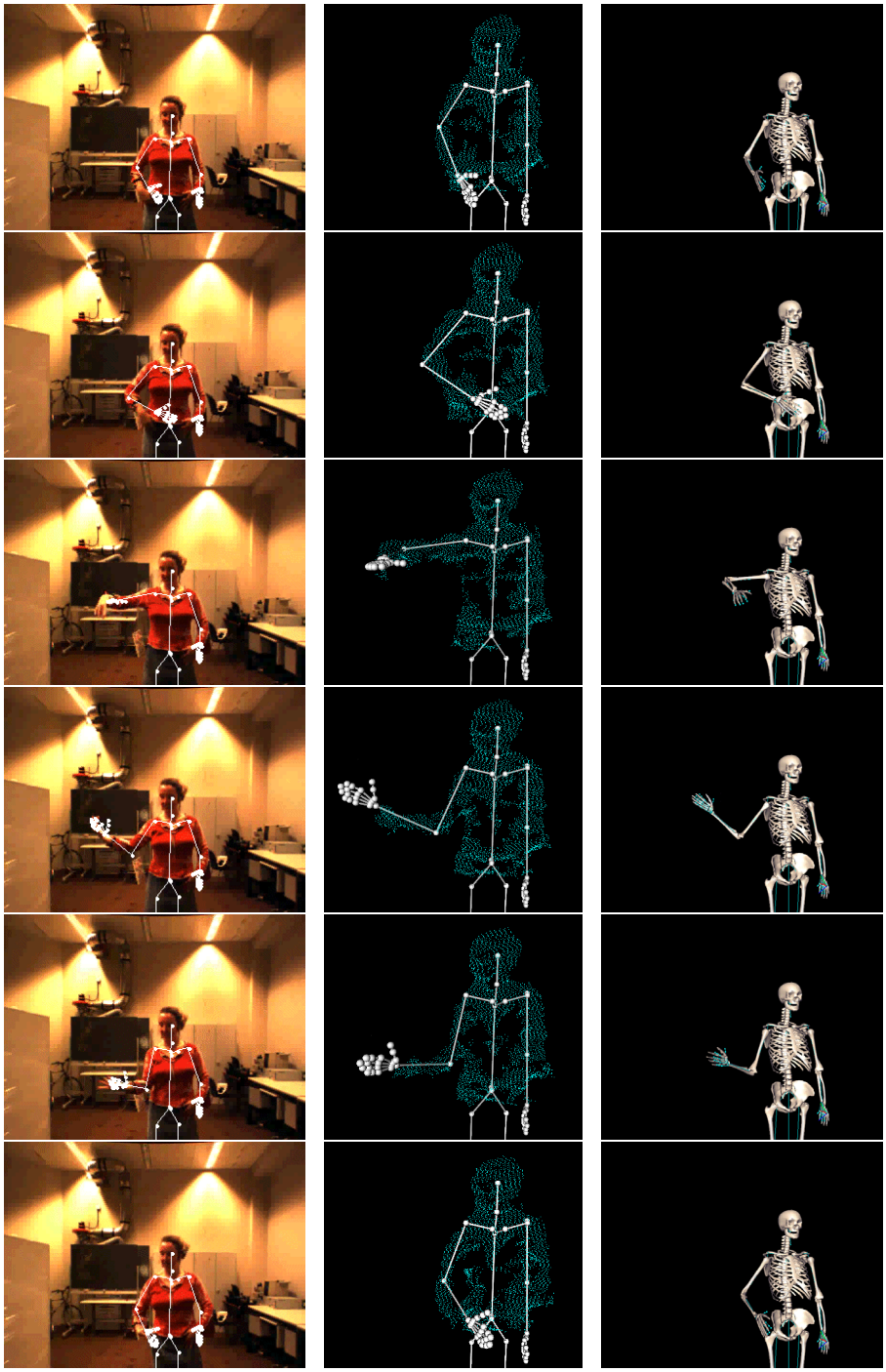


Figure 5.42: Unconstrained tracking of sequence 3 (1). The featured frames are numbered 1, 15, 28, 33, 35 and 40.

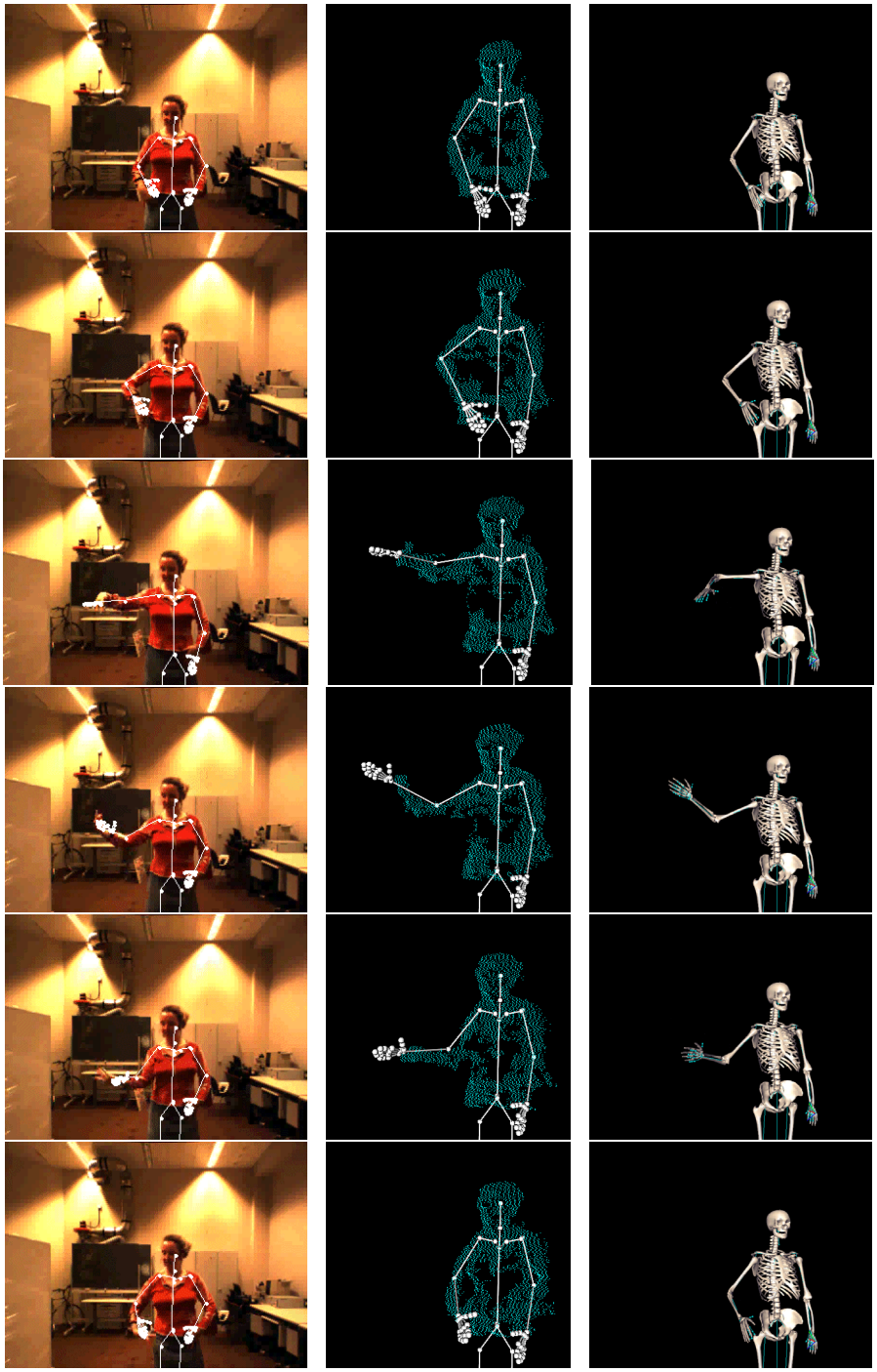


Figure 5.43: Constrained tracking of sequence 3 (1).

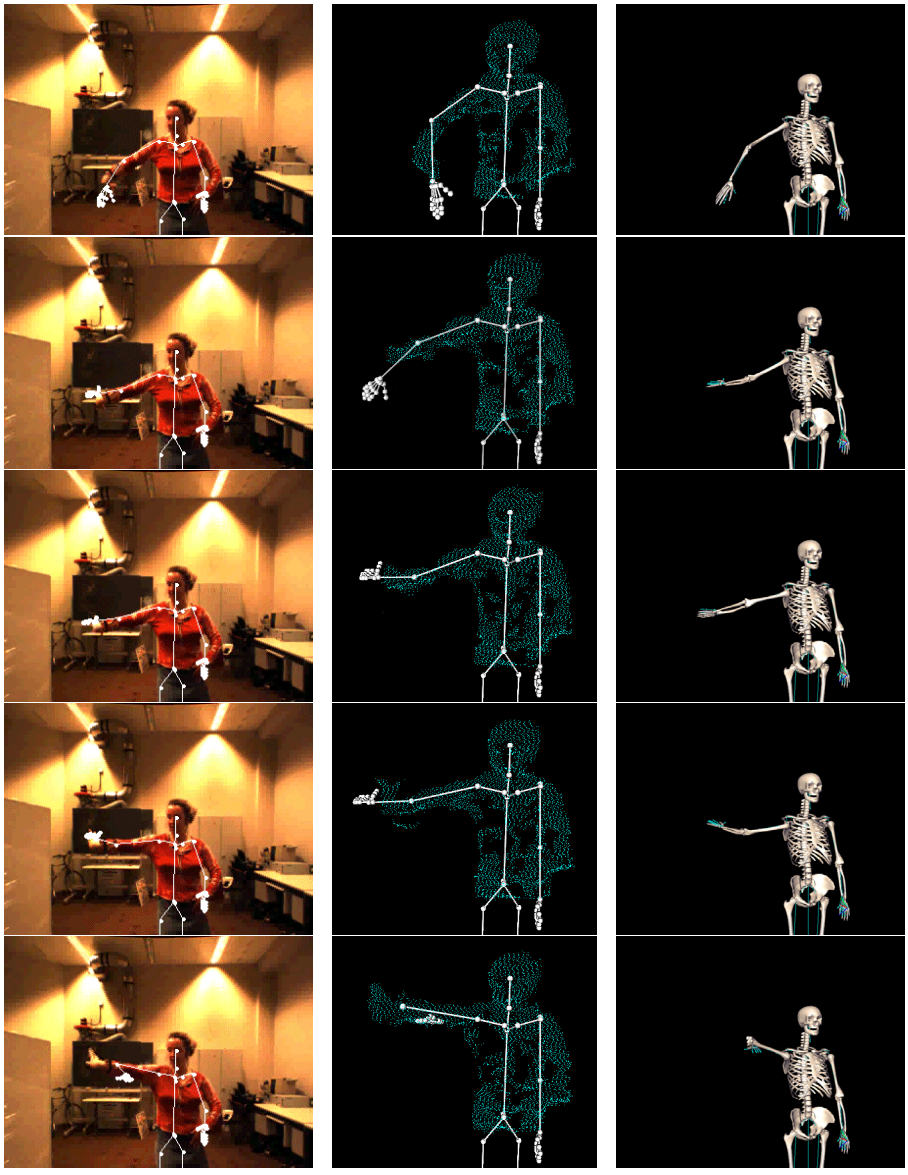


Figure 5.44: Unconstrained tracking of sequence 3 (2). The featured frames are numbered 45, 48, 49, 50 and 51.

Figure 5.45: Constrained tracking of sequence 3 (2).

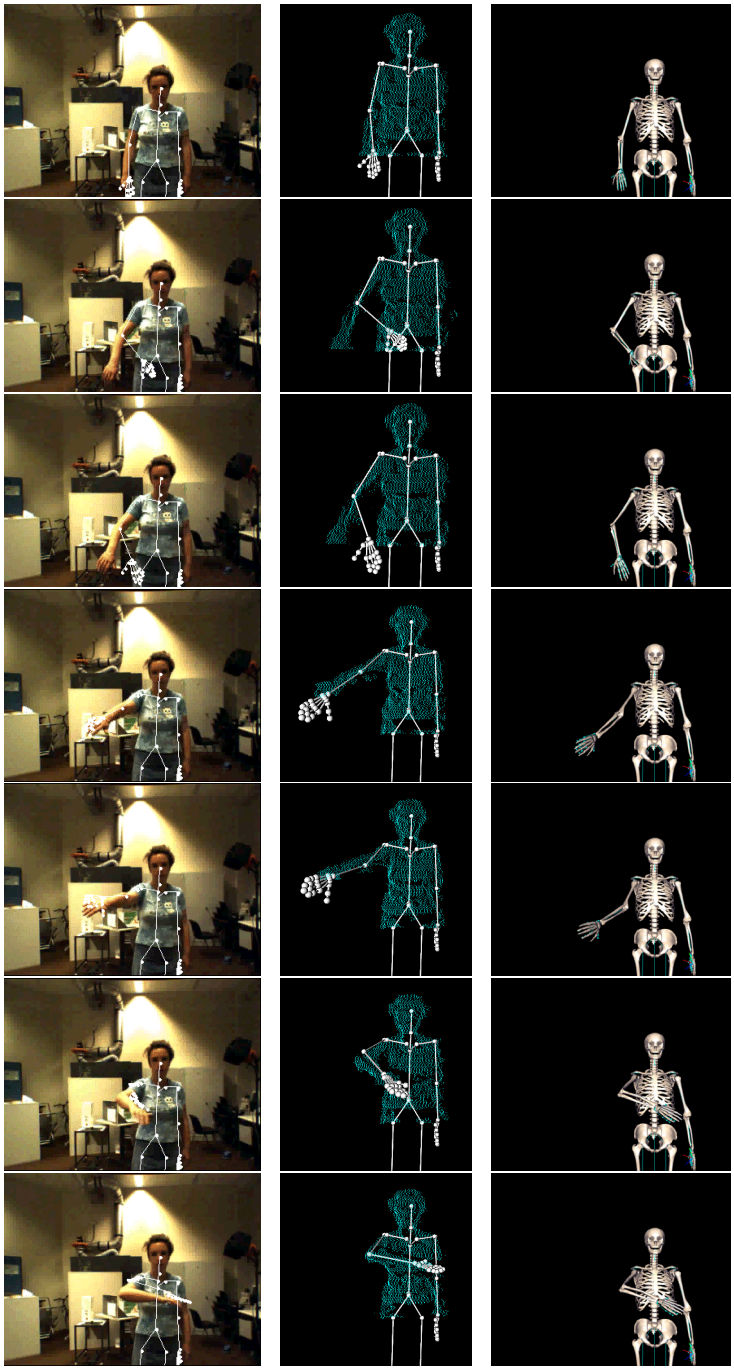


Figure 5.46: Unconstrained tracking of sequence 4. The featured frames are numbered 1, 23, 25, 31, 34, 46 and 52.

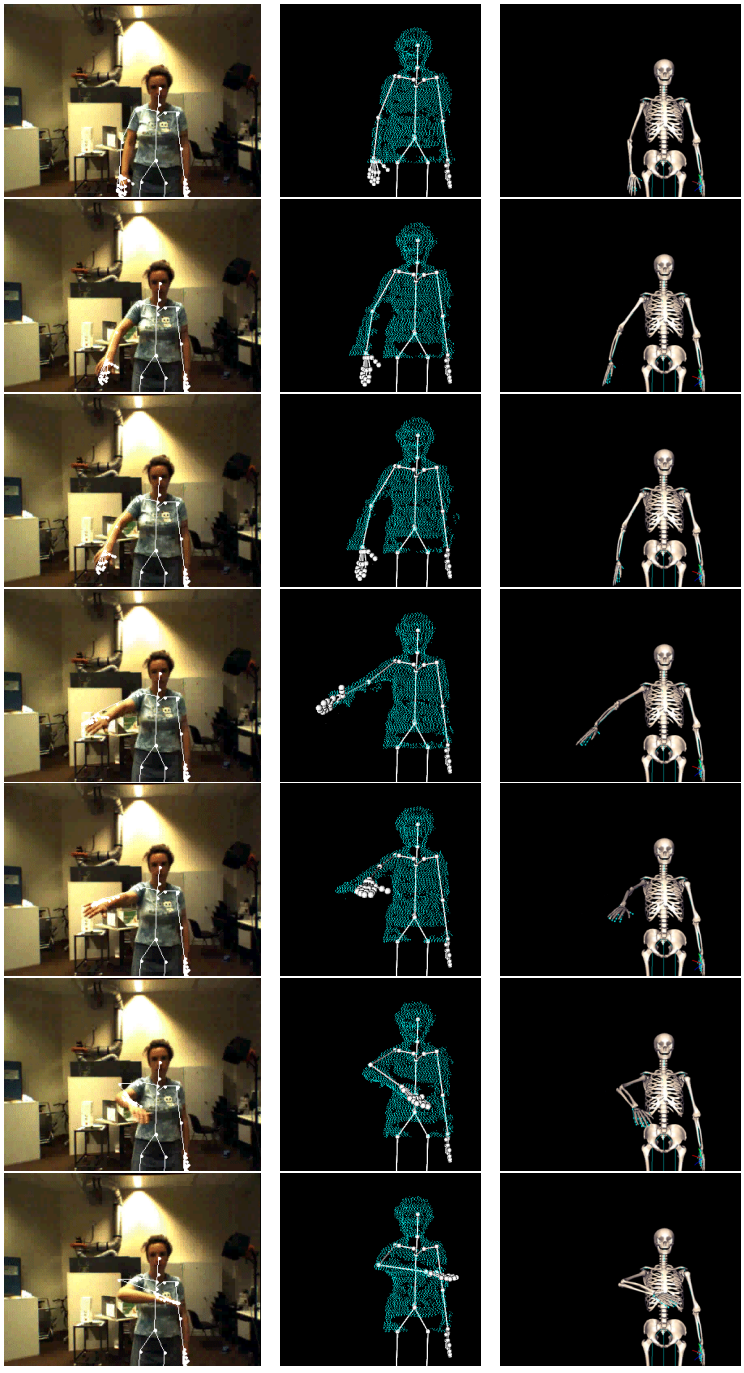


Figure 5.47: Constrained tracking of sequence 4.

5.4.2 Hierarchical sterno-clavicular and gleno-humeral joint limits

5.4.2.1 Character animation

The result of using sterno-clavicular and gleno-humeral joint limits to constrain motion are shown in the following pages. The contribution of the sterno-clavicular joint to upper limb motion is visible on two levels:

- arm abduction: independently from clavicular motion, the gleno-humeral joint has an abduction range of roughly $10\,\mathrm{deg}$. For the arm to be further abducted, the sterno-clavicular joint needs to be set in motion. The two joints contribute to arm elevation at a ratio of 2-to-1, for the gleno-humeral and sterno-clavicular joint respectively [Otani1989]. In the lower hemisphere, i.e. for adduction, the clavicula does not act as a limiting factor.
- arm flexion and extension: as in the abduction case, the ranges of flexion and extension of the gleno-humeral joint are limited without contribution from the clavicula, to a smaller extent as for abduction, however.

Figures 5.48 and 5.49: in the first frame of Figure 5.48, we show the maximal range of abduction available at the gleno-humeral joint for the initial pose of the sterno-clavicular joint (approximately $8\,\mathrm{deg}$ of abduction), this range being limited to $17\,\mathrm{deg}$. In the second frame, we further abduct the sterno-clavicular joint, yielding an equivalent increase in abduction for the gleno-humeral joint, as shown in the third frame. A second increase in abduction is shown in the fourth and fifth frames. In the sixth frame, we re-set the extension amount of the sterno-clavicular joint to its original value and in the first frame of Figure 5.49 show the corresponding extension range of the gleno-humeral joint. Increasing the extension value of the sterno-clavicular joint, as shown in the second frame, leads to an increased extension range of the gleno-humeral joint, as shown in the third frame. Finally, in the fourth frame, we show the initial amount of flexion available at the gleno-humeral joint for the initial position of the sterno-clavicular joint. In the fifth frame, we flex the sterno-clavicular joint some more, resulting in the full flexion of the upper arm, as shown in the last frame.

The joint limits were also applied to the keyframe sequence of Figure 5.30 that we used previously for the gleno-humeral and elbow joint limits.

Figures 5.50 and 5.51: this sequence was modified by key-framing in order to add exaggerated sterno-clavicular joint motion. In the left column is displayed the original out of boundary motion, and in the right column, the corrected motion using hierarchical sterno-clavicular and gleno-humeral joint limits. Note that no joint limits are applied to the elbow.

5.4.2.2 Motion capture

We captured a sequence using the Digiclops^{TM} system to obtain stereo data, and fitted our body model to the data, optimising the degrees of freedom of the sterno-clavicular and gleno-humeral joints.

A very simple motion such as gradual raising of the arm is sufficient to high-light problems that occur when coupling is ignored between the two joints. The resulting errors may seem unimportant but if the recovered postures were then imported into an animation software and skinning applied, we would obtain erroneous skin deformations such as those displayed in Figure 3.11. The errors obtained in terms of incorrect clavicula elevation are shown in Figure 5.52 for the video sequence that is partially shown on the following pages.

This sequence displays, in Figure 5.53, the results of tracking the stereo data without applying any joint limits. The observed behaviour is contrary to the anatomically-correct one. In the first frame of the sequence, the posture seems relatively correct but closer inspection of the sterno-clavicular rotational values reveals that the clavicula is subject to slightly too much abduction, the values however remaining within the valid range. As the arm is progressively raised, the total amount of abduction of the upper arm should be distributed among the sterno-clavicular and gleno-humeral joints. However, this is not what occurs. On the contrary, the greater the abduction at the gleno-humeral level, the smaller the abduction at the sterno-clavicular joint. When the arm is almost vertical, in frame 136, the clavicula is by then nearly horizontal, when in fact it should have reached its

Figure 5.48: Sterno-clavicular and gleno-humeral motion (1).

maximum abduction. When applying the coupled joint limits, we obtain the tracking results displayed in Figure 5.54, in which we observe that the clavicula does contribute its share of upper arm abduction, as it should.

Thanks to the contribution of our hierarchical joint limits, the animator designing a sequence can be absolutely certain that the created poses are perfectly valid in terms of anatomical joint limits. Furthermore, keyframe sequences that were designed without the use of precise joint limits can be corrected by simply running them through a keyframe reader that supports hierarchical joint limits and computes the validated postures. In terms of motion capture, the increased quality of the output postures is undeniable, as it allows to discard postures that are not physically possible. This is especially vital in the event of an over-determined solution space, as was the case for optical motion capture, and in the presence of extremely noisy data, such as the real data used for our examples, where outliers cause great confusion. That joint limits are not useless even for motion capture

Figure 5.49: Sterno-clavicular and gleno-humeral motion (2).

on synthetic 3D data was also proven. Even with perfect data, the recovery of motion at the extremities accepts a vast array of solutions due to the fact that any amount of axial rotation will explain the data, as was the case for optical motion capture. Ideally, to maximise the pruning of the search-space, both extrinsic and intrinsic constraints should be applied simultaneously. As we pointed out in Section 1.4.2, our aim was not to propose a complete motion capture framework, but to prove the usefulness of applying constraints in terms of posture recovery. The technical details as to how the hierarchical joint limits were derived, from rotation measurement up to the final morphing step, follow in the upcoming chapter.

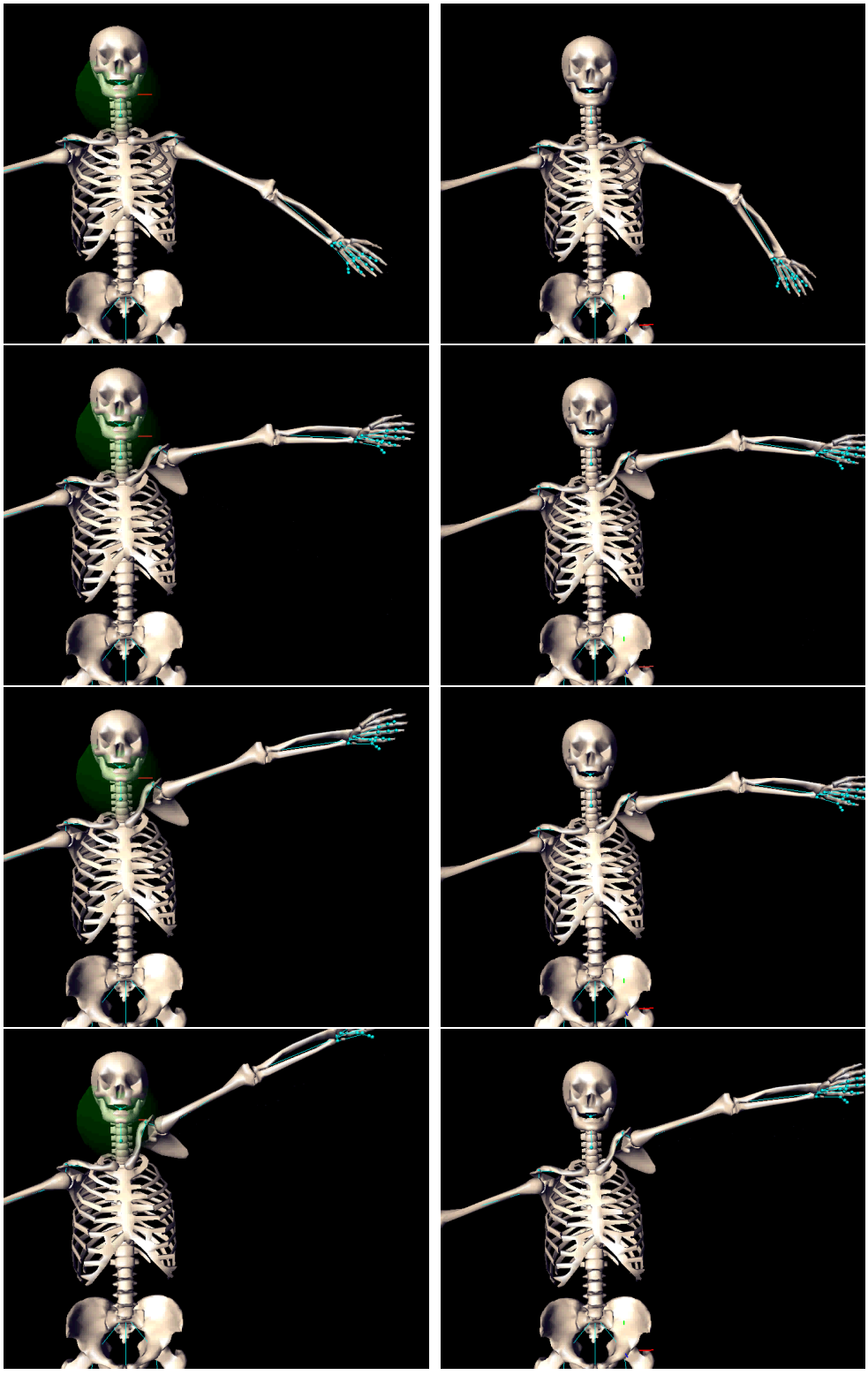


Figure 5.50: Frames of the “Are you crazy?” sequence with sterno-clavicular and gleno-humeral joint limits (part 1).

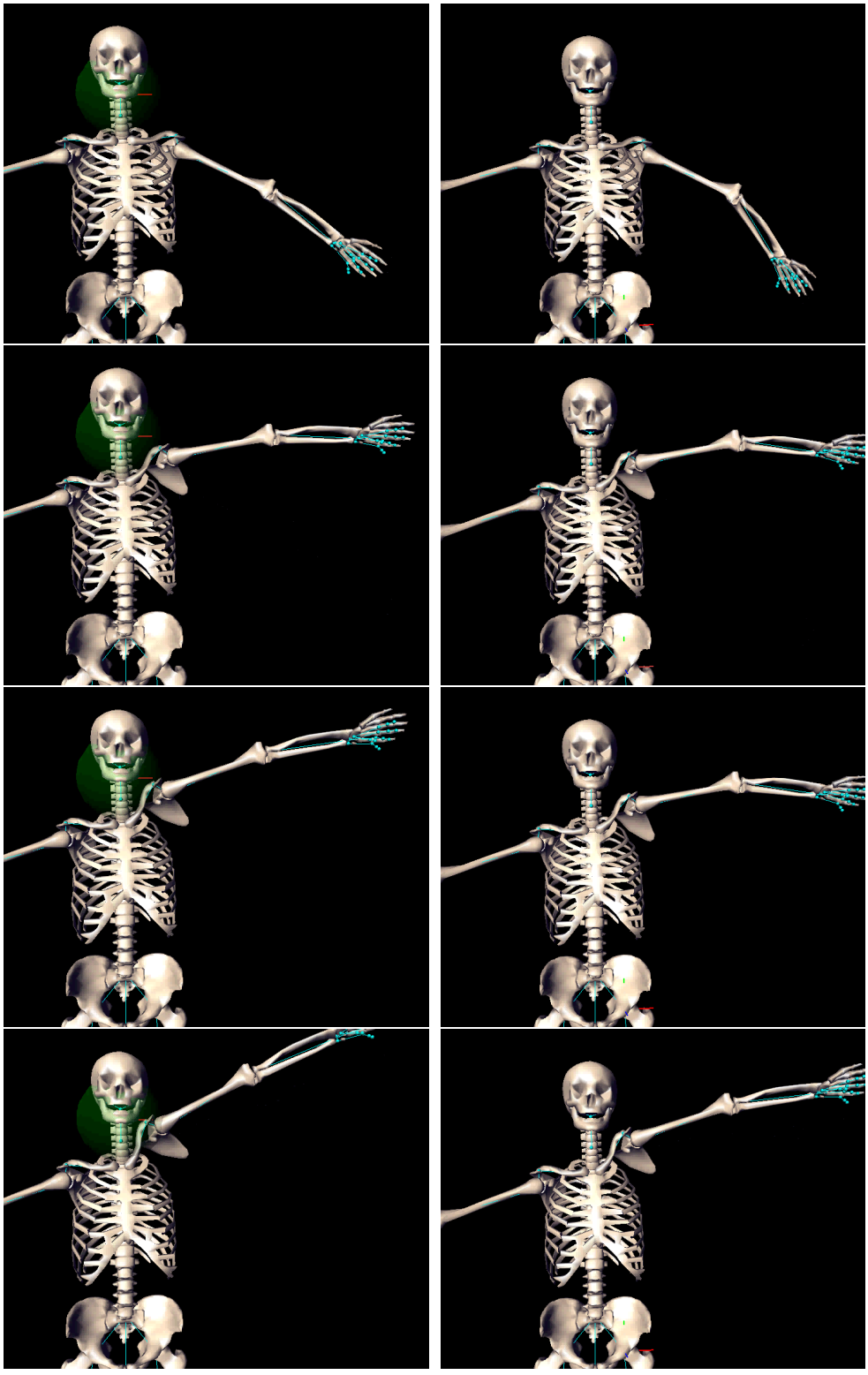
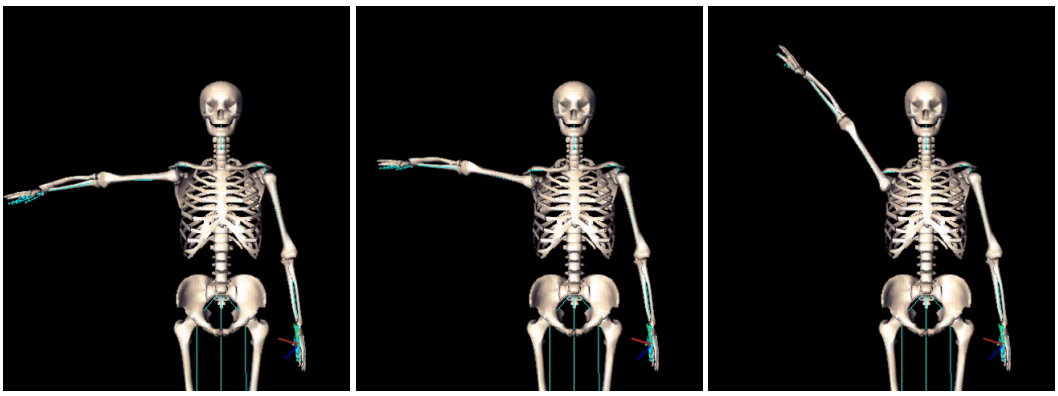


Figure 5.51: Previous sequence continued (part 2).



(a) (b) (c)

Figure 5.52: Postures corresponding to frames of the tracked arm-raising sequence without joint limits: (a) at frame 76, the arm is almost horizontally outstretched at this point, the clavicula should have abducted slightly but as one can see, it is still in the same position as if the arm were resting at the side; (b) in frame 84, the clavicula actually adducts as the arm is raised which is contrary to the expected behaviour; (c) in frame 136, the arm is close to the vertical position and the clavicula should at this point have reached its maximum abduction value, which is 40 degrees. However, it has further adducted so as to be perfectly horizontal, a position that is physically impossible.

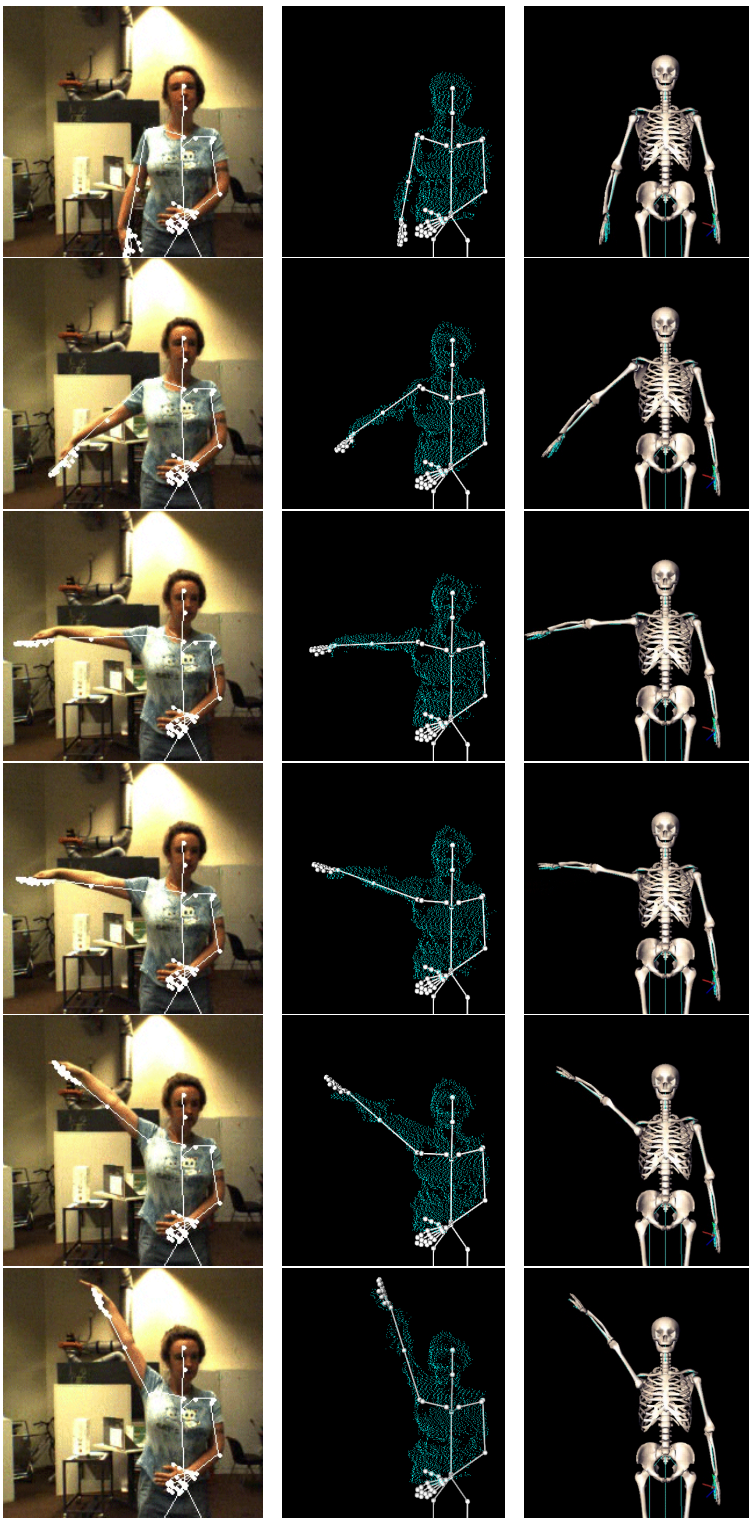


Figure 5.53: Unconstrained tracking for the arm-raising sequence.

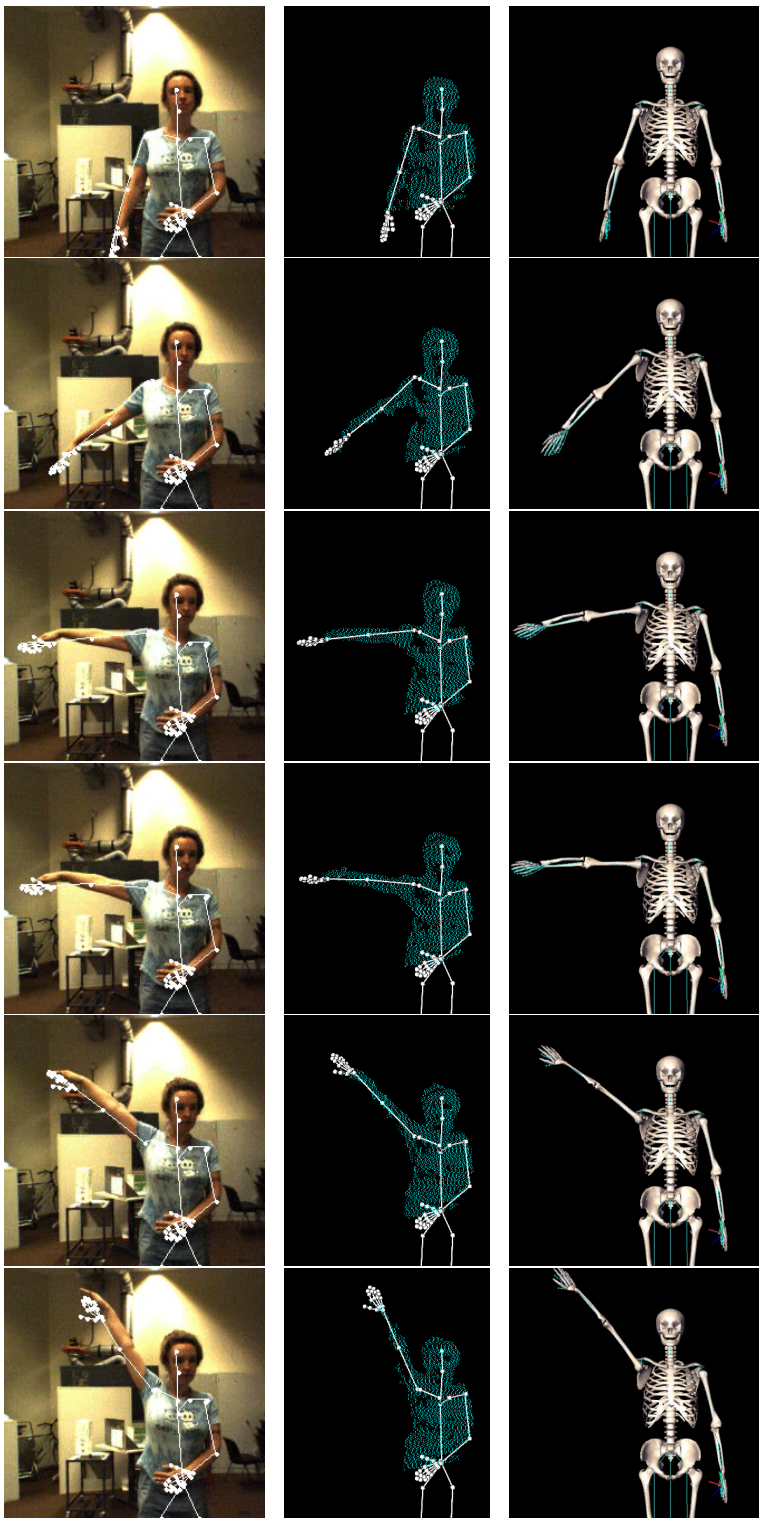


Figure 5.54: Constrained tracking.

Chapter 6

Instantiating the joint limits representation

Having briefly laid out in the previous chapter the joint limits formalism we have defined, we will now describe in detail the various steps involved in achieving those results. In order to compute realistic joint limits for a given joint, the *modus operandi* can be summarised in three main steps:

1. measuring the range of motion of the targeted joints and collecting the corresponding data;
2. converting the measured data to a usable 3D rotation representation;
3. extracting an envelope for these rotations, thus defining a boundary of valid rotations.

As we intend our formalism to be universal, we need to be able to readily compute joint limits for any joint in the human body. As there is a certain number of them, and even if most of them do not necessarily need to be used for motion capture, this should be feasible within a decent amount of time. Overall conditions should therefore be that:

- joint rotation values should be relatively easy and fast to collect. In other words, the measurement protocol should not be so constraining that data collection is painstaking and/or requires the subject to be deceased. Also, the acquisition system should allow the collection of a large amount of data within a short time frame;
- the entire process from data collection to a joint limits representation should be fast, readily repeatable, and should involve minimal user intervention.

We believe that our method complies with all these conditions, and in the following sections, we will take the reader through the main steps that are involved.

As in the previous chapter, we will be taking a coarse-to-fine approach, starting with considering upper arm motion to be defined by a ball-and-socket joint that actually comprises the motion of the entire shoulder compound. By using unit quaternions to represent rotations, we obtain compact data in a space that has a meaningful distance metric. These unit quaternions are approximated by an implicit surface envelope, whose analytical formulation allows us to easily distinguish valid rotations from invalid ones. In Section 5.3.2, we described how the single 3 DOF joint limits approach could be extended to a coupled set of joints, by defining clusters of similar rotations of a the parent joint, and thus deriving joint limits for the child joint for each such cluster. This method was applied to the sterno-clavicular and gleno-humeral coupled joints, as well as the gleno-humeral and elbow coupled joints. The first set defines the coupling of a 2 DOF parent joint with a 3 DOF child

joint, and the second set, a 3 DOF parent joint with a 2D child joint. The rotations of each joint are determined in quaternion space, due to the necessity of having a distance metric in 3D, but the 2 DOF joint rotations can be converted to their equivalent Euler angle rotation shortly before implicit surface approximation. The reason for this is that the motion of two specific joints is continuous in 2D. The implicit surface is thus fitted to quaternion data in the case of a 3 DOF joint and to Euler angle data in the case of a 2 DOF joint. The hierarchical joint limits for coupled joints are derived by clustering the parent joint data, this directly providing the corresponding child joint rotations for each cluster. The latter are approximated in turn by an implicit surface, thus defining different joint limits for different clusters. The clustering can be refined by computing intermediate clusters and their child joint limits using a morphing scheme that will be detailed at the end of this section.

6.1 Motion measurement

Measuring the amount of rotation that is available at each human joint is obviously a matter of great interest to researchers in the fields of biomechanics, sports studies, physiotherapy, clinical studies and biomedical engineering. Studying the biomechanical behaviour of the human body is helpful not only for pure research's sake, but also for understanding pre- and post-surgery, rehabilitation, pathologies, sports performance and more.

The shoulder compound, due to its complexity, is a particular tough subject to tackle, not only because of its multiple linkages, but also due to the fact that a greater part of its motion factors are invisible from the surface. The human upper limb as a whole has also been the subject of measurement attempts, in order to define the workspace of the arm. Depending on the degree of exactitude that is required, more or less precise measures have been carried out, the most recent and significant ones of which are reviewed in the upcoming state-of-the-art sub-section. The section that immediately follows describes our own measurement protocol for collecting data corresponding to the range of motion of the shoulder compound, but also to the coupled ranges of motion of the sterno-clavicular and gleno-humeral joints, and the gleno-humeral and elbow joints.

6.1.1 Upper limb measurement to this day

The range of motion of the shoulder complex and of the joints that compose it has been measured over and over again, and published in a certain number of biomechanical journals and conferences. In the computer graphics and vision area, the few joint limit representations that exist are exclusively based on the available clinical data .

With the exception of Engin and Tümer, none of the other research groups' ultimate aim was to find a usable joint limits representation, this therefore posing the problem of how to derive a working range of motion formalism from their collected data.

[Engin and Chen1986] published a widely-used statistical database, but their data unfortunately concerns only global upper arm angular motion. [Engin and Tümer1989a] then researched the angular range of motion of each individual joint within the shoulder complex. These are however expressed specifically in terms of Euler angles, suppose a local co-ordinate system identical to their own, and disregard the axial rotation component. Their measurement protocol consisted of a torso-restraining system, along with an arm cuff, both of which prevent torso motion and arm twisting that would otherwise falsify the validity of the measures. Sonic emitters were used to derive the locations and orientations of the upper arm, and optimisation techniques applied to the data to infer the rotational values of the linkages of the shoulder complex model they defined.

The Delft shoulder group aimed at designing an animated model of the shoulder compound, and to this end, the ranges of motion of all the shoulder complex joints were measured. In order for this to be feasible, and to obtain 'real' data, various methods were applied. To one exception, namely the use of an electro-magnetic motion capture device, in [Meskers *et al.*1999], bony landmark palpation methods were the rule. In this manner, they gathered a set of data explaining the functioning of each individual joint with a great deal of precision, as in [VanDerHelm and Pronk1995, Klein Breteler *et al.*1999, DeGroot1997]. Given the fact that each such landmark needs to be digitised one at a time, for each posture, the amount of collected data is relatively small, and would therefore not allow us to derive a compact space for the ranges of motion. Furthermore, they admit that the use

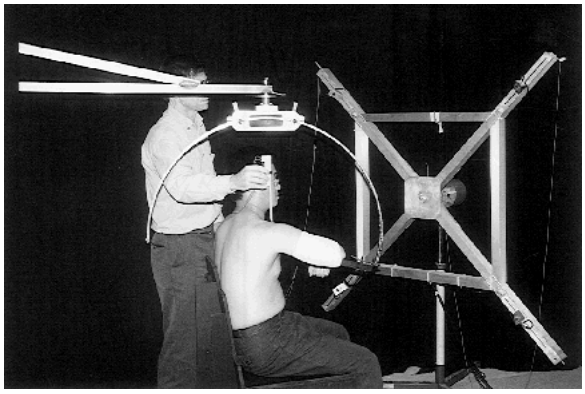


Figure 6.1: Experimental set-up for upper arm rotations with sonic digitiser [Wang *et al.*1998].

of Euler angles to express 3D rotations has serious drawbacks, as we ourselves pointed out in the chapter dealing with body models, namely that:

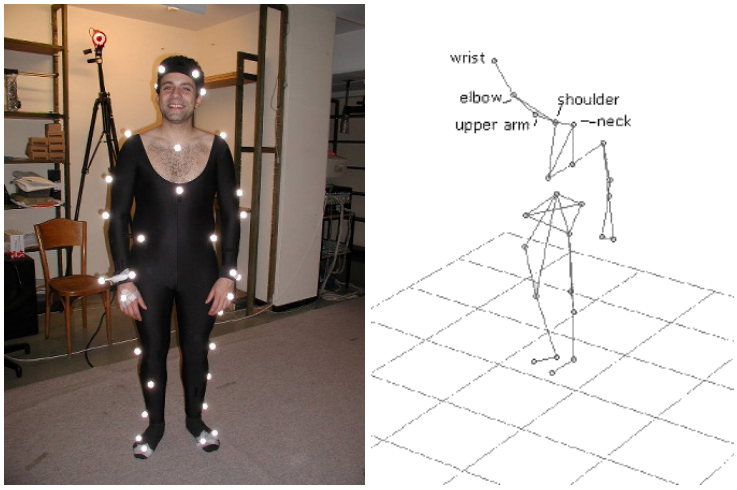
- the Euler angle representation for a given rotation is not unique, forcing a smoothing technique to be applied to ensure that the chosen set of values is the 'closest' in terms of the single-axis values with respect to the previous 3D rotation;
- that some measured data needs to be discarded due to the Gimbal lock effect that blocks one degree of rotational freedom.

[Högfors *et al.*1991] published their findings in terms of rotations measured on each shoulder complex joint, expressing these rotation as Euler angles in global 3D space, for the clavicular, scapular and humeral bones. Their measurement technique is not minimally invasive in the least, as it requires the percutaneous insertion of tantalum balls into the bones of the subjects, fortunately under local anaesthesia. These balls can then be located during motion using X-ray exposures. The resulting data for the various subjects shows, however, very little cohesion, so no clear motion range was derived from it. Furthermore, their measurement protocol is not readily applicable in just any computer vision laboratory that does not employ a surgeon, and we will therefore not consider it for our purposes.

[Wang *et al.*1998] used a device similar to the one of Engin and Chen to track upper arm motion (Figure 6.1), where the torso was fixed, and the arm was kept flexed in a cast. A sonic digitiser connected to a computer allowed the recovery surface landmark positions. This of course implies the same drawback as the Delft Shoulder Group palpation method, namely that the amount of data collected was limited due to the tediousness of the process. On the basis of 40 postures recorded for a set of seven subjects, they derived the range of angular and axial motion of the upper arm, but unfortunately not of the individual shoulder girdle joints. [Biryukova *et al.*2000] also measured upper arm motion, using an electro-magnetic device, thus inferring the range of motion of the shoulder complex as a whole.

[VanDerHelm1997] published a measurement protocol for the various joints in the shoulder complex on the basis of palpation method, and suggested that it would be extendible to video-based data collection methods. However, he did specify that in the absence of body landmarks or visible motion for some of the rotational components, one would need to use linear regression to derive the parameters that cannot be directly measured. This is namely the case for all scapula rotation, and for axial rotation of the clavicular bone.

As none of the data in the literature was re-usable and/or dense enough to be able to derive a joint limits representation from it that would include all motion components, we propose to measure the rotational data ourselves. Encouraged by Van der Helm's statement that measurement was possible even using video-based methods, we decided to use optical motion capture to collect the necessary data.



(a) (b)

Figure 6.2: Vicon Motion Capture Session: (a) actor with markers; (b) reconstructed 3D markers.

6.1.2 Optical motion capture measurement

Optical motion capture provides us with a very convenient tool, in the sense that data collection is non-invasive, relatively simple, and can be carried out at a high frequency, thus guaranteeing a considerable amount of data within a short capture time. For the readers who were attentive during the section dealing with 3D marker position reconstruction in optical motion capture, they will remember that it is not error-free, and that a small degree of imprecision remains. The increasing number of motion capture companies who propose products specially dedicated to human motion analysis for medical applications reassures us as to the precision of the marker reconstruction. The main source of noise is, as always with surface markers, the relative motion of the skin with respect to the underlying bones, which we can unfortunately in no way remedy.

6.1.2.1 Motion measurement

We have captured joint motion using the Vicon^{TM} Motion Capture System, the capture volume being surrounded by eight cameras to reconstruct the 3D positions of the reflective markers, whose diameter is approximately 1 cm. Reconstruction is precise and in the absence of occlusion, no markers are lost in general. Automatic labelling is sometimes erroneous over some frames, or ambiguous, due to the fact that only inter-marker distances are used to identify them, these distances being computed during calibration of the motion capture subject, which consists of a pre-defined initialisation pose.

The reflective markers are placed strategically on the actor, either on a tight suit, or directly on the skin. Their locations vary depending on the joint motion we intend to measure. Figure 6.2(a) shows the actor wearing the marker set for complete body motion.

In all motion capture cases on the upper body, an additional marker is placed at neck level to serve as a fixed reference.

If we wish our joint limits to be as precise as possible, and to reflect the range of motion as closely as possible, we need to pay attention to sampling the space of attainable postures not only homogeneously, but also densely. In general, such a capture sequence lasts several minutes, so that we may obtain such a data set.

The Vicon motion capture system then reconstructs the markers in 3D global space for every image frame of the sequence, and labels those that it was able to identify. This enables us to retrieve the global 3D positions of

our markers over the entire sequence, accessing their co-ordinates by the names of the markers.

6.1.2.2 Rotation computation

Motion representation The obtained set of possible joint orientations and positions in space can be considered as a path of referential frames in 3D space [Bloomenthal1990], which can then be mapped to a sub-space of the space of possible quaternion referential frames of the object, called the Quaternion Gauss map [Hanson1998a]. In other words, paths in the space of possible referential frames correspond to quaternion referential frames on this map. A very complete overview of the concept can be found in [Hanson1998b].

Quaternion field technology Our goal is to create an analysable topographic space of joint orientations using the experimental 3D marker co-ordinates as the basis. To accomplish this, we adopt the method of quaternion fields described in [Hanson1998a] and [Hanson and Ma1995]. In the quaternion field method, each joint orientation is first converted to a 3×3 matrix M, where, using Euler's theorem[1], M may be expressed in terms of its lone real eigenvector $\hat{\mathbf{n}}$ and the angle of rotation θ about that axis. This in turn may be expressed as a point in quaternion space, or, equivalently, a point on a three-sphere S^3 embedded in a Euclidean 4D space. To find θ and axis $\hat{\mathbf{n}}$, given *any* rotation matrix or frame M, we need two steps:

1. Solve for θ using
$$\mathrm{Tr}M = 1 + 2\cos\theta \tag{6.1}$$
2. Find $\hat{\mathbf{n}}$ from
$$M - M^t = \begin{bmatrix} 0 & -2n_3\sin\theta & 2n_2\sin\theta \\ 2n_3\sin\theta & 0 & -2n_1\sin\theta \\ -2n_2\sin\theta & 2n_1\sin\theta & 0 \end{bmatrix} \tag{6.2}$$
3. which is defined as long as $\theta \neq 0$. In the degenerate case, the matrix and the quaternion are taken to be the identity transformation. For more details, see [Biedenharn and Louck1981].

The identification of the corresponding quaternion follows immediately from

$$q(\theta, \hat{\mathbf{n}}) = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2}) \tag{6.3}$$

up to the sign ambiguity between the two equivalent quaternions q or $-q$, which correspond to the exact same rotation matrix M. A single compact volume containing a dense set of quaternion frames will have two distinct but equivalent realisations due to the $q \Leftrightarrow -q$ ambiguity. We have implemented a procedure to iteratively check the 4D dot-product between neighbouring quaternions and force all neighbours to have signs that are consistent with positive dot products.

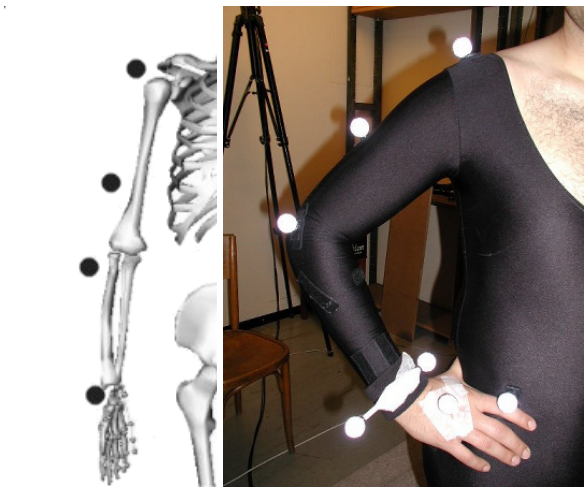
6.1.2.3 Measuring the range of motion of the shoulder compound

We will now describe the process involved in capturing the motions of the shoulder compound considered as a unique ball-and-socket joint.

For this purpose, we take into consideration the four markers positioned on the upper limb as depicted by Figure 6.3. The markers are placed on the shoulder joint, upper arm segment, elbow and inner side of the wrist, and their Vicon labels are respectively SHO, UPA, ELB and WRA, plus the neck marker.

To generate a dense data set that covers the entire range of possible shoulder motions, we captured several sequences during which the subject attempted to perform a complete as possible set of motions such as the one depicted by Figure 6.4.

[1] Not to be confused with Euler angles! Euler's theorem states that any 3D rotation can be expressed as a rotation of magnitude θ around an axis $\hat{\mathbf{n}}$.



(a) (b)

Figure 6.3: Marker positioning: (a) relative position of the markers with respect to the skeleton; (b) actual motion capture markers on an actor.

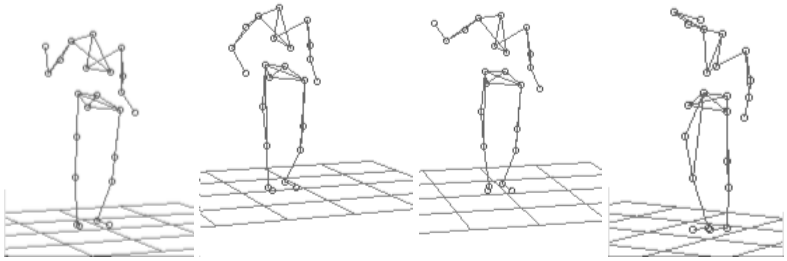


Figure 6.4: Shoulder motion sequence.

The measurement protocol contains some special requirements that need to be heeded by the actor, as these conditions not being fulfilled would cause axial rotation computation to be erroneous:

1. the elbow joint should always be slightly flexed, as the SHO, ELB and WRA markers would otherwise lie on a line and not on a plane;
2. no twisting should be performed at any moment at the lower arm level, as this would rotate the WRA marker, and thus rotate the previously-mentioned plane.

Construction of shoulder co-ordinate frames To build our frames from the marker co-ordinates, we begin by substracting the neck marker co-ordinates from the other four co-ordinates, so that all co-ordinates are expressed with respect to a fixed point.

The frames are then constructed as follows: the first axis of the frame corresponds to the upper arm segment, i.e., the line defined by the shoulder (SHO) and upper arm (UPA) markers. The second axis is the normal to the plane passing through the shoulder (SHO), elbow (ELB) and wrist (WRA) markers (this plane representing the axial rotation - or "twist" motion - of the shoulder joint). The third axis is simply orthogonal to the other two, i.e., their cross-product. The construction of the referential is illustrated in Figure 6.5.

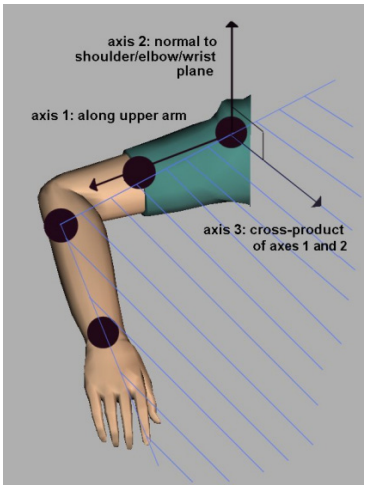


Figure 6.5: Deriving a referential from the markers.

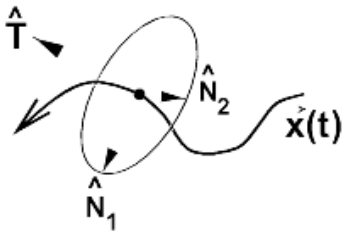


Figure 6.6: General form of a moving frame for a 3-D curve $\mathbf{x}(t)$, with the tangent direction $\hat{\mathbf{T}}$ determined directly from the curve derivative, and the exact orientation of the basis $(\hat{\mathbf{N}}_1,\ \hat{\mathbf{N}}_2)$ for the normal plane determined only up to an axial rotation about $\hat{\mathbf{T}}$.

An orthonormal co-ordinate frame is created in this manner for each image frame of the sequence, along with the rotation relating one image frame to the next. The calculated 3×3 rotation matrices $M(t)$ for each data point are then converted into unit quaternions as described above, producing a dense cloud of points occupying a volumetric portion of the quaternion 3-manifold (the three-sphere S^3). The concept of a generic moving frame on a curve is illustrated by Figure 6.6.

When transformed to quaternion form, each successive frame from the moving sequence becomes a point on a quaternion curve lying within the three-sphere S^3 embedded in R^4. Families of frames on a surface become surfaces in quaternion space, and collections or disjoint sets of orientations such as those collected here for the shoulder joint become a cloud of points occupying a volumetric quaternion space with a distinct closed boundary surface.

By retaining only the spatial co-ordinates (q_x, q_y, q_z) of the quaternion, we can represent the four motions of Figure 6.4 as trajectories in 3D space. By processing in this fashion the long motion capture sequence, we obtain the cloud of 3D points depicted by Figure 6.7(a) and (b).

6.1.2.4 Measuring the coupled ranges of motion of the gleno-humeral and elbow joints

If we now want to measure the simultaneous motion of two joints, say the gleno-humeral and elbow joints, we need to define a new measurement protocol.

Indeed, the marker configuration of the previous sub-section depicted by Figure 6.5 becomes insufficient,

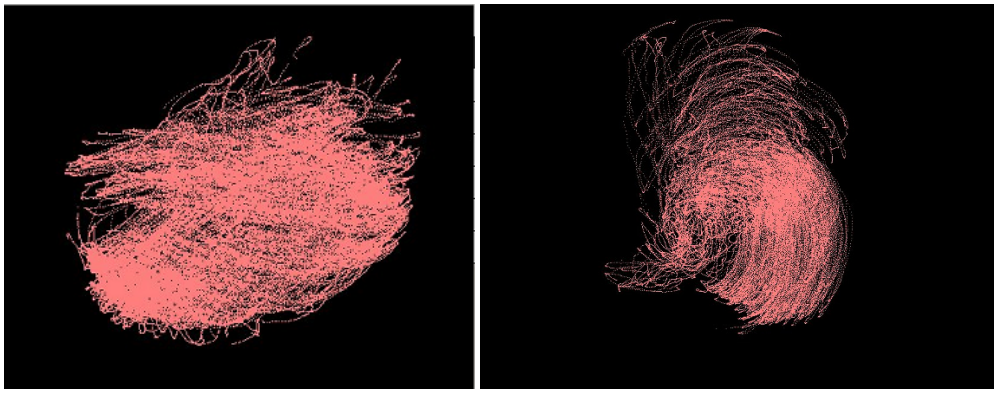


Figure 6.7: Volume of 3D quaternion data points from two different view points.

as we need extra markers on the elbow to measure its axial rotation. Furthermore, we can longer use the plane formed by the shoulder (SHO), elbow (ELB) and inner wrist (WRA) markers to infer axial rotation for the shoulder, as this plane will now be rotating due to elbow axial rotation.

We therefore have to perform new motion capture sessions specifically for the gleno-humeral and elbow joints. As shown in Figure 6.8, we keep the shoulder, upper arm, elbow and inner wrist markers as before, and add to this set a marker on the lower arm, close to the elbow, across from the ELB marker. This will be referred to as the forearm marker (FRM). The ELB and FRM markers are those that are hardly affected by twisting of the lower arm. We also now take into consideration the outer wrist marker (WRB).

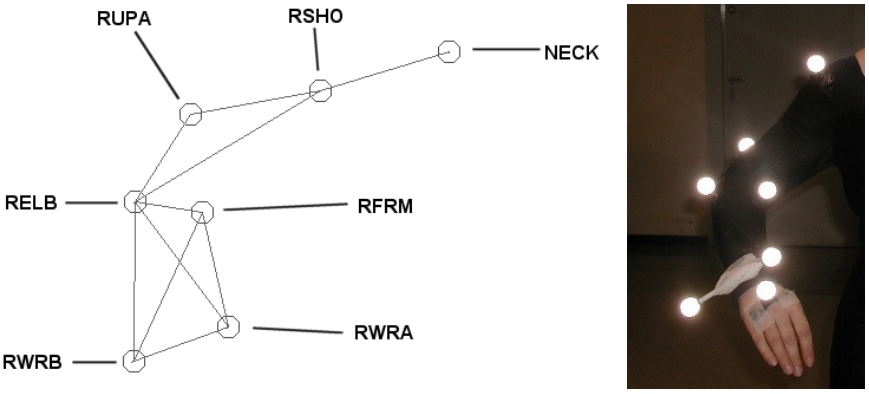
The motion capture actor is less constrained as in the shoulder compound measurement session, as he no longer needs to pay special attention to keeping the elbow flexed and the wrist static. However, to ensure that we obtain homogeneous sampling for both joints, the recommended protocol is to place the upper arm at various elevations in the reachable space, and then to perform a slight twist of the upper arm. At each such position, the actor should then completely flex and extend the lower arm, as well as twist the lower arm as far as possible in both directions.

Computing the rotations We construct the rotating frames of the gleno-humeral joint as follows: the first axis of the frame corresponds as before to the upper arm segment, i.e., the line defined by the shoulder (SHO) and upper arm (UPA) markers. The second axis is the normal to the triangle whose vertices are the upper arm marker (UPA), the elbow marker (ELB) and the forearm marker (FRM). This is the plane that represents axial rotation. The third axis is again orthogonal to the other two, i.e., their cross-product.

The constructed frames will then not only provide us with the gleno-humeral joint rotations, but they are also the local joint referentials, in which all elbow joint rotations will be expressed.

As all marker positions are in 3D global space, in order to obtain elbow joint rotations, we first must transform all marker positions from global referential to local gleno-humeral joint referential. This is simply done by applying the constructed frames to the 3D marker co-ordinate vectors. We can now construct the rotation frames for the elbow joint. For this, we use:

- the upper arm segment,
- the lower arm segment $(lowerarm_x, lowerarm_y, lowerarm_z)$ defined as the line between the elbow (ELB) and outer wrist (WRB) markers,
- the elbow segment $(elbow_x, elbow_y, elbow_z)$, i.e. the line defined by the elbow (ELB) and forearm (FRM) markers, and
- the wrist segment $(wrist_x, wrist_y, wrist_z)$, i.e. the line defined by the inner (WRA) and outer (WRB) wrist markers.



(a) (b)

Figure 6.8: Motion capture measurement protocol for the gleno-humeral and elbow joints: (a) placement of the markers on the shoulder, upper and lower arm; (b) actual motion capture actor with markers.

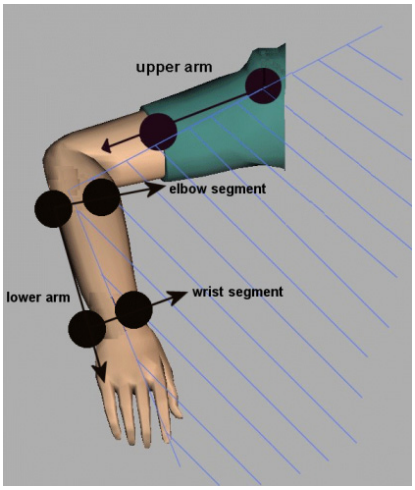


Figure 6.9: Marker positions and derived segments and axes for computing the various rotations.

The segments so defined are shown in Figure 6.9.

As the elbow joint is only 2D with one swing component and a twist component, we do not construct an orthogonal frame. Instead we derive flexion and twist separately, due to the fact that converting a 3×3 matrix to a Euler angle would in no way guarantee it to have one zero swing component.

We start by computing what we call an offset matrix, representing the initial offset between the lower arm and elbow segments, i.e initial elbow flexion. Logically, we could have directly computed the angle between lower arm and upper arm segments to derive flexion, but this yields numerical instability around the value $\frac{\Pi}{2}$, so the flexion values need to be evaluated incrementally on the basis of the initial offset. This offset matrix mat_{offset} is computed by dividing matrix m_2 by matrix m_1 as per:

$$m_1 = \begin{bmatrix} \overrightarrow{x_1} \\ \overrightarrow{y_1} \\ \overrightarrow{z_1} \end{bmatrix}, \overrightarrow{x_1} = \begin{bmatrix} lowerarm_x \\ lowerarm_y \\ lowerarm_z \end{bmatrix}, \overrightarrow{plane_{dir}} = \begin{bmatrix} elbow_x \\ elbow_y \\ elbow_z \end{bmatrix}, \overrightarrow{y_1} = \overrightarrow{plane_{dir}} \wedge \overrightarrow{x_1}, \overrightarrow{z_1} = \overrightarrow{x_1} \wedge \overrightarrow{y_1} \quad (6.4)$$

$$m_2 = \begin{bmatrix} \overrightarrow{x_2} \\ \overrightarrow{y_2} \\ \overrightarrow{z_2} \end{bmatrix}, \overrightarrow{x_2} = \begin{bmatrix} -elbow_y \\ elbow_x \\ elbow_z \end{bmatrix}, \overrightarrow{y_2} = \overrightarrow{plane_{dir}} \wedge \overrightarrow{x_2}, \overrightarrow{z_2} = \overrightarrow{x_2} \wedge \overrightarrow{y_2}$$

where $\overrightarrow{x_2}$ is the vector orthogonal to the initial elbow segment.

As we wish to obtain a unique value corresponding to pure flexion, we need to ensure that the two other rotation components are nil. If we simply convert mat_{offset} to its equivalent Euler angle values, we will most likely obtain a Euler angle with three non-zero components, due to the approximate nature of the data, and due to the non-unique conversion from a rotation matrix to a Euler angle, as discussed in the Appendix.

We therefore apply least-squares fitting procedure to obtain a 3D Euler angle with two zero components. To this end, our matrix mat_{offset} is converted to a quaternion, which is then stored as the original value of rotation, q_{init}. We revert to quaternions at this point because they provide us with a measure of similarity for 3D rotations, which the Euler angle formalism does not. Such a metric is vital to our enterprise as we need to ensure that we do not drift too far away from the original quaternion q_{init}. We create a new quaternion with two zero components, keeping only the component corresponding to flexion, yielding in this case $q = (0., q_y, 0.)$. We are thus optimising only one parameter, though we need to ensure at all times that the quaternion remains unitary, meaning that q_y should remain within the $[-1., 1.]$ interval. To this end, we have implemented a penalty function f:

$$f = \begin{cases} -x - 1 & \text{if } q_y \in]-\infty, -1] \\ x - 1 & \text{if } q_y \in [1, \infty[\\ 0 & \text{elsewhere} \end{cases}$$

with gradient g:

$$g = \begin{cases} -1. & \text{if } q_y \in]-\infty, -1] \\ 1. & \text{if } q_y \in [1, \infty[\\ 0 & \text{elsewhere} \end{cases}$$

For computing the objective function with respect to the current value of q_y, we compute the quaternion

$$q_{diff} = q - q_{init}$$

If the norm of q_{diff} is close to zero, then we need not optimise any further, as this means that we have found a one-zero component equivalent quaternion that is extremely close to q_{init}. In the other case, the optimiser needs to iterate, and we therefore estimate the values of the gradient g by computing the divided differences for the current value q. We define:

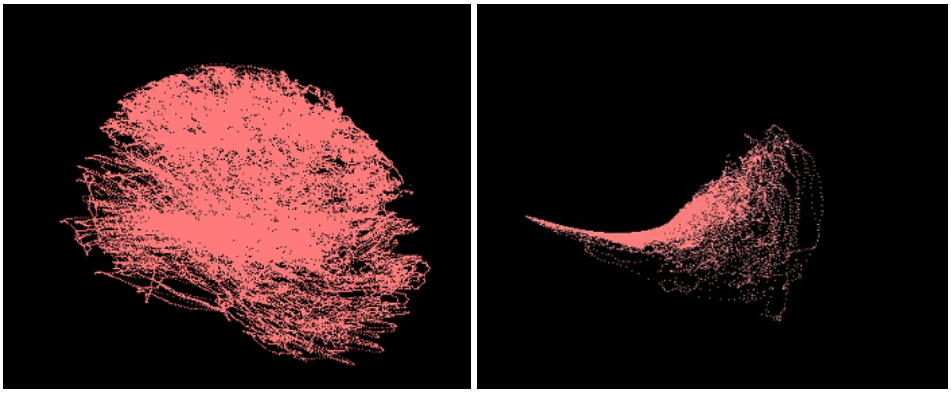
$$q_\Delta = (0., q_y + \Delta, 0.)$$

where Δ is a very small value and estimate the objective functions for q and q_Δ:

$$f = ||q - q_{init}|| \ \ and \ \ f_\Delta = ||q_\Delta - q_{init}||$$

so g becomes:

$$g = \frac{f - f_\Delta}{\Delta}, \quad (6.5)$$



(a) (b)

Figure 6.10: 3D quaternion data from the coupled motion capture of the gleno-humeral and elbow joints: (a) gleno-humeral joint data; (b) elbow joint data in the gleno-humeral joint referential.

The scalar component of the resulting quaternion is set to positive and computed as per eq.(5.2). Converting this quaternion to its equivalent Euler angle, we thus obtain the offset value corresponding to initial elbow flexion. The computation steps for extracting the flexion and twist values from the marker data are detailed in the paragraphs below.

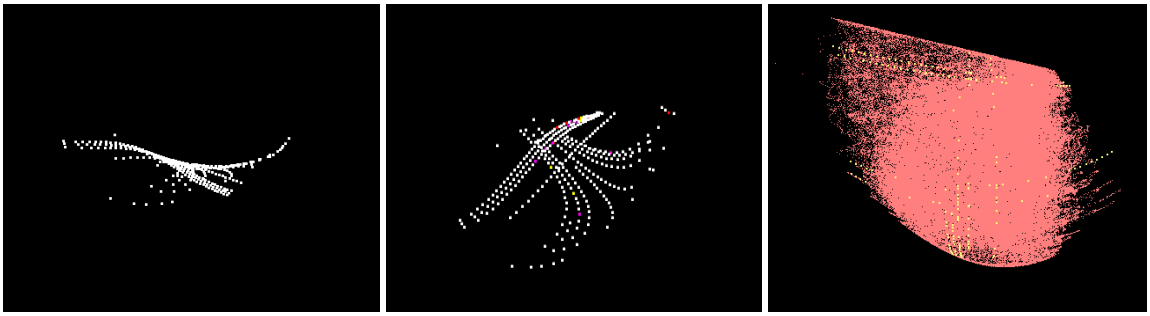
Flexion Flexion could simply be derived from the angle between the lower and upper arm segments, but this is not possible, as the lower arm segment experiences axial rotation when the elbow is twisted. Indeed, elbow twisting (forearm pronation/supination) in a fixed upper arm referential modifies the position of the lower arm axis that we use for measuring flexion. This is reported by [Biryukova *et al.*2000], who state that the position of this axis can vary from $11\,\mathrm{deg}$ to $36\,\mathrm{deg}$ depending on the subject.

To avoid the influence of the forearm position on the measured flexion component, we will once again use the vector orthogonal to the elbow segment. We then compute the difference between the initial flexion offset and the current position, by estimating the rotation matrix that transforms the position of the elbow segment in the previous frame to its position in the current frame. For a given frame *f*, the matrix m_f equals:

$$m_f = \begin{bmatrix} \overrightarrow{x} \\ \overrightarrow{y} \\ \overrightarrow{z} \end{bmatrix}, \overrightarrow{x} = \begin{bmatrix} -elbow_y \\ elbow_x \\ elbow_z \end{bmatrix}, \overrightarrow{plane_{dir}} = \begin{bmatrix} elbow_x \\ elbow_y \\ elbow_z \end{bmatrix}, \overrightarrow{y} = \overrightarrow{plane_{dir}} \wedge \overrightarrow{x}, \overrightarrow{z} = \overrightarrow{x} \wedge \overrightarrow{y}$$

The rotation matrix is then $\frac{m_{f-1}}{m_f}$ where m_{f-1} is the matrix of the previous frame. We perform least-squares optimisation on this matrix, as we did for the offset matrix, and by converting the quaternion result to its equivalent Euler angle, we obtain the relative flexion value. If this is the first frame, then the flexion value is the offset value plus the relative value, in all other cases, we just add the relative value to the flexion value of the previous frame.

Twist To derive the value of axial rotation of the lower arm, we need to compute the rotation that leads from the elbow segment to the wrist segment, the latter being rotated by the twisting motion, and the elbow segment serving as fixed reference. The matrix m_1 corresponding to the elbow segment referential is defined as per eq.(6.4). The matrix for the wrist segment is defined in the following manner:



(a) (b) (c)

Figure 6.11: 3D quaternion data for the elbow joint originating from a simulated motion capture session: (a) side view; (b) top view; (c) superposition of the simulated and real data.

$$m_2 = \begin{bmatrix} \overrightarrow{x_2} \\ \overrightarrow{y_2} \\ \overrightarrow{z_2} \end{bmatrix}, \overrightarrow{x_2} = \overrightarrow{x_1}, \overrightarrow{plane_{dir_2}} = \begin{bmatrix} wrist_x \\ wrist_y \\ wrist_z \end{bmatrix}, \overrightarrow{y_2} = \overrightarrow{plane_{dir_2}} \wedge \overrightarrow{x_2}, \overrightarrow{z_2} = \overrightarrow{x_2} \wedge \overrightarrow{y_2}$$

The axial rotation is obtained by dividing m_1 by m_2. However, as in the case of flexion, we cannot simply convert this matrix to a Euler angle and hope to obtain a pure twist value. This is due to the fact that during motion capture, we cannot ensure that the elbow and wrist segment planes of rotation remain perfectly parallel at all times, which would be a *sine qua non* condition in order to obtain a pure twist component from the matrix. To overcome this problem, we perform least-squares optimisation on this matrix, in order to find the closest quaternion where all components except twist are nil. Converting the resulting quaternion to a Euler angle, we obtain our twist value.

Applying these procedures to our measured data, we obtain the 3D quaternion data for the gleno-humeral and elbow joints shown in Figure 6.10(a) and (b) respectively.

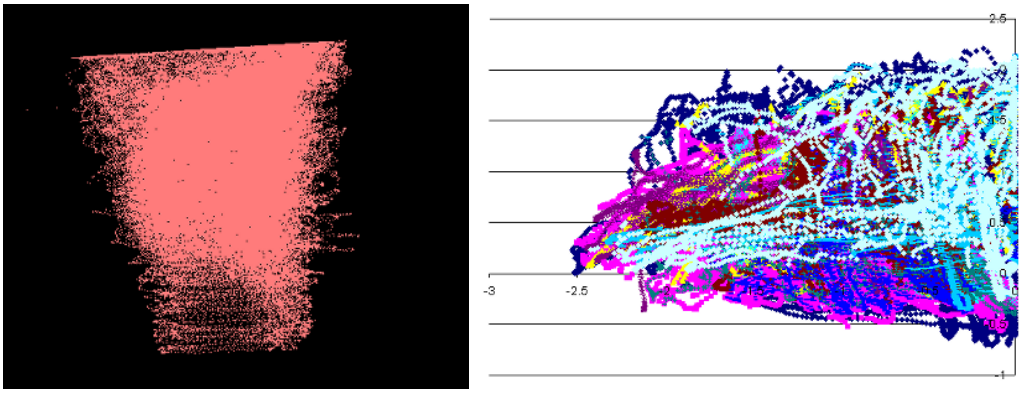
To check whether the general shape of our elbow quaternion is correct, we perform a simulation in our application and obtain the data shown in Figure 6.11.

For a ball-and-socket joint, it is of course useful to express joint limits in quaternions, as was pointed out in Section 5.3. However, in the elbow joint case, this no longer holds true, as the joint rotation has components in only two planes and can therefore without risk be expressed as a Euler angle rotation around two axes only. The problem of Gimbal lock no longer is one. Furthermore, we now have a 3D quaternion cloud for a 2D joint, which is not exactly a simplification. Therefore we convert our cloud of quaternion data of Figure 6.10(b) to Euler angles and obtain the new data shown in Figure 6.12(a). A plot of the data on two axes is shown in 6.12(b), where the horizontal axis is flexion, and the vertical axis is twist.

6.1.2.5 Measuring the coupled ranges of motion of the sterno-clavicular and gleno-humeral joints

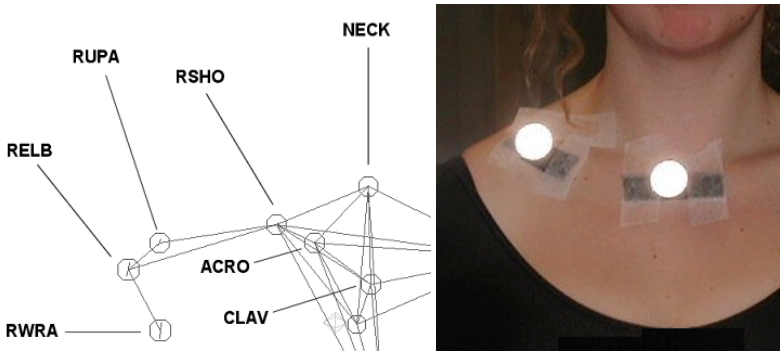
Having computed the rotations for the coupled gleno-humeral and elbow joint set, we will now complete our upper limb model by also inferring the rotations for the previous coupled joint set in the hierarchy, namely the sterno-clavicular and gleno-humeral joints.

For measuring the rotations of the gleno-humeral joint, we can re-use either one of the measurement protocols outlined in the two previous sub-sections.



(a) (b)

Figure 6.12: Euler angle data for the elbow joint: (a) cloud of data; (b) plot of flexion vs. twist values.



(a) (b)

Figure 6.13: Motion capture measurement protocol for the sterno-clavicular and gleno-humeral joints: (a) placement of the markers on the clavicula, shoulder and arm; (b) additional markers on motion capture actor.

For measuring the motion of the clavicular bone, we place a marker at either end of it, at the visible bony landmarks. The first one is on the sterno-clavicular joint (CLAV), and the second one on the acromio-clavicular joint (ACRO), as shown in Figure 6.13(b).

No special constraint is placed on the motions of the actor, but as before, the aim was to homogeneously sample the range of motion of both joints, so the subject was to perform upper arm abduction/adduction, flexion/extension and twist at various clavicle elevations.

Computing the rotations As the elbow joint, the sterno-clavicular joint performs motion in only two planes, so we could retrieve these motion components separately without needing to construct a referential frame. However, as we then need to express gleno-humeral motion in the sterno-clavicular referential, the construction of a local referential frame is nevertheless necessary.

It is the motion of the clavicular bone that we are attempting to reconstruct, therefore, we compute the co-ordinates of the clavicular segment $clavicula$ corresponding to this bone as the segment between the sterno-clavicular (CLAV) and acromio-clavicular (ACRO) markers. We construct a referential frame around this seg-

Figure 6.14: Assessing the drift between the marker placed on the acromio-clavicular bony landmark and actual position of the joint, for the case of motion in the coronal plane.

ment in the following fashion:

$$m_{ref} = \begin{bmatrix} \overrightarrow{x} \\ \overrightarrow{y} \\ \overrightarrow{z} \end{bmatrix}, \overrightarrow{x} = \begin{bmatrix} clavicula_x \\ clavicula_y \\ clavicula_z \end{bmatrix}, \overrightarrow{plane_{dir}} = \begin{bmatrix} clavicula_x \\ clavicula_z \\ -clavicula_y \end{bmatrix}, \overrightarrow{y} = \overrightarrow{plane_{dir}} \wedge \overrightarrow{x}, \overrightarrow{z} = \overrightarrow{x} \wedge \overrightarrow{y}$$

The sterno-clavicular rotation has two swing components (abduction/adduction and flexion/extension) that are computed in a similar fashion as flexion and twist in Section 6.1.2.2, for the gleno-humeral and elbow joints.

We then apply the constructed referential frames of the clavicula to the 3D marker co-ordinate vectors, and construct the gleno-humeral rotation frames.

As the global position of the motion capture subject is static, we measure the amount of angular rotation with respect to the horizontal vector corresponding to an outstretched arm. We compute the rotation matrix that performs the rotation of this vector the the positions of the clavicular bone.

As the sterno-clavicular joint is mobile only in two planes, we need to obtain a Euler angle with one zero component. As before, this is not simply obtained by converting the rotation matrix to its equivalent Euler angles.

We therefore again apply a least-squares fitting procedure to obtain a 3D Euler angle with one zero component. We proceed as in Section 6.1.2.2 for the gleno-humeral and elbow joints, computing the quaternion representing the difference between the initial value and the current value, the value of the objective function being the norm of this quaternion. The gradient for the flexion component is computed as in eq.(6.5), and the abduction component in a similar manner, by varying the corresponding parameter in the divided differences expression. The resulting Euler angle will have one nil component, its two other components being the abduction and flexion values.

Skin-to-bone displacement As [VanDerHelm1997] pointed out, measuring motion of the clavicular bone is difficult using non-intrusive measurement techniques. Indeed, the bone is salient and clearly visible when the

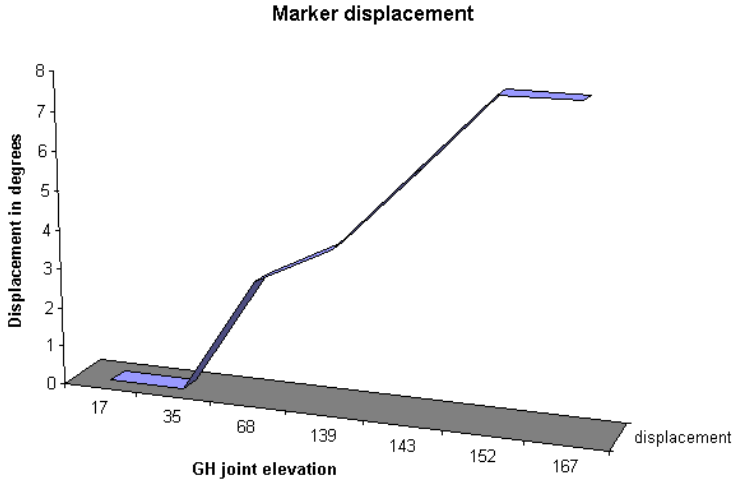


Figure 6.15: Displacement of the marker with respect to its underlying acromio-clavicular joint, for upper arm abduction.

arm is alongside the body. At a certain amount of arm abduction, the relative motion of the soft tissues and skin with respect to the clavicular bone become large and the acromio-clavicular joint can no longer be precisely located with purely optical methods. Indeed, we observe:

- the increasing displacement of the skin with respect to the acromio-clavicular joint during arm elevation;
- the reduced displacement of the skin with respect to the clavicular bone during sterno-clavicular flexion.

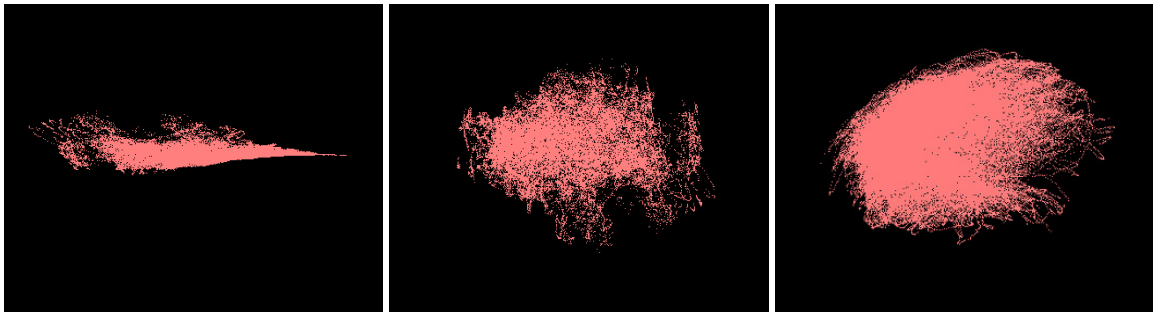
In practical terms, this means that if a marker is placed at the acromio-clavicular joint to track the extremity of the clavicular bone, this marker attached to the skin will drift further and further away from the actual position of its underlying joint during abduction of the clavicula. We observe the reversed trend in the case of flexion of the clavicula, namely that the underlying joint is displaced by a larger amount than the skin itself. This phenomenon was fortunately minimal in our case, as no major discrepancy was noted between our computed flexion values and those reported in the literature.

However, the abduction values we have computed for the sterno-clavicular joint need to be adjusted in order to take into account the relative skin-to-bone motion. To this effect, we have measured this displacement error by locating the markers in a video sequence, and the bony landmarks using palpation. In this manner, we can measure the drift between the marker and the actual acromio-clavicular joint position in the coronal plane (see Figure 6.14).

We note that the displacement occurs as soon as the upper arm is abducted by about 50 degrees starting from the arm along the body. From there on, the displacement is linear as depicted by the graph in Figure 6.15. This error function is integrated into our sterno-clavicular motion computation, so as to obtain an approximation of the actual values of abduction. By proceeding in this manner, we obtain the 3D quaternion data for the sterno-clavicular and gleno-humeral joints in Figure 6.16.

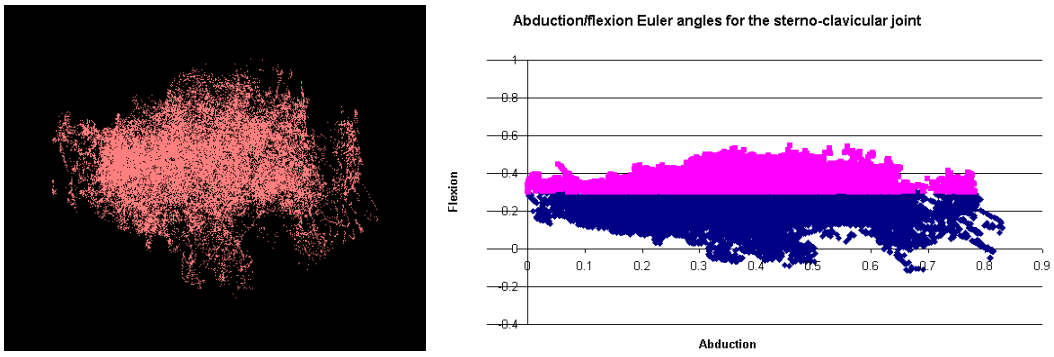
As in the case of the elbow joint, we transform the sterno-clavicular joint quaternions into Euler angles with one zero component. We obtain the resulting cloud of data of Figure 6.17.

By strategically placing markers on the upper limb and around the shoulder area, we are able to derive the corresponding joint rotations from the 3D marker trajectories. By converting the resulting rotations into unit quaternions, we obtain dense and compact data sets representing the joint ranges of motion, the quaternion space metric providing us with a measure of similarity between the rotation data points. For the joints having motion components in one plane only, such as the sterno-clavicular and elbow joints, we ultimately converted our quaternions to Euler angles, as in this manner we obtain 2D data whose boundary will be easier to determine



(a) (b) (c)

Figure 6.16: 3D quaternion data from the coupled motion capture of the sterno-clavicular and gleno-humeral joints: (a) sterno-clavicular joint data, side view; (b) top view; (c) gleno-humeral joint data.



(a) (b)

Figure 6.17: (a) Euler angle data for the sterno-clavicular joint, (b) Plot of abduction vs. flexion values.

then in its unit quaternion form. This conversion is acceptable due to the fact that the two afore-mentioned joints do not present inter-joint dependencies.

6.2 Computing implicit surface joint limits

To make effective use of the collected data, we now need to derive a working representation for it. In the 3D quaternion sub-space (q_x, q_y, q_z), we will approximate the cloud of points representing all the valid measured rotations for a single joint by some kind of envelope that will enable us to decide whether a movement is possible or not, using an inside/outside test. As discussed in the previous chapter, we have settled on an implicit surface representation because it has the advantages of being smooth and continuous, the interior or exterior of this surface being elegantly determined by its analytical formulation.

However, as we have seen from the data, it neither forms a clean contour nor surface, so we would first need to extract the contour or surface points in order to be able to wrap an implicit surface around them. Experience has shown us that the nature of our data does not readily allow this, and in the end, we had to approach the problem directly from a volumetric aspect, which nevertheless enabled us to succeed in meeting our target. Our various and vain attempts at extracting the surface points are related in the first sub-section, as they very clearly

illustrate the inapplicability of existing methods to our range of motion data.

The implicit surface we extracted using our volumetric approach, described in the second sub-section, suffers from needless complexity, and we will therefore endeavour to render it less so, which is the purpose of the third sub-section. Finally, we will extend our technique to the case of a set of coupled joints, deriving a hierarchical representation to capture the inter-dependence.

6.2.1 Deriving surface points from volumetric rotational data

When 3D data is extremely dense, homogeneous and convex, extracting surface points is easy. However, even though we wish our data had such characteristics, it unfortunately does not. A human being is subject to feelings of discomfort and pain, and will therefore be more willing to perform some shoulder motions than others, regardless of the clear instructions (s)he may have received. The absence of any interactive tool that would allow immediate feed-back during motion capture makes it impossible to be aware of such phenomena on the spur of the moment. No matter how many motion capture sessions we carried out, we ended up with the same trends in over-population and desolation within the ranges of motions.

The fact that our data is therefore unevenly dense, especially so at the borders, makes our life hard in terms of surface point recovery. Various methods have been put to the test, in order to identify the one that would works best.

6.2.1.1 Solid angles

To turn 3D points into a surface, we first tried a method based on dividing the 3D space into overlapping solid angles placed at the centre of mass of the cloud. To this end, the 3D Cartesian points $X_j(x, y, z)$ are converted into spherical co-ordinates $X_j(\rho, \theta, \phi)$. The 3D space is then divided into solid angles by taking into account the points that fall within the intervals $[\theta, \theta + d\theta]$, $[\phi, \phi + d\phi]$.

Let $SP(i, k)$ be the points inside the (i, j) solid angle, then $\rho_{max}(i, j)$ is defined as the maximum radius of the (i, j) solid angle points.

$$\rho_{max}(i,k) = \max_{x_j \in SP(i,k)} \rho_j$$

The surface points are taken to be those whose radius is close to ρ. For each solid angle, we write

$$Surface(SP(i,k)) = \{x_j, x_j(\rho_j, \theta_j, \phi_j) \in SP(i,k) \cup (1-T)\rho_{max}(i,k) < \rho_j \leq \rho_{max}(i,k)\}$$

where T is an appropriate threshold value. The complete surface becomes the union of all the surface points

$$Surface(SP) = \bigcup_{i=1}^{L} \bigcup_{k=1}^{L} Surface(SP(i,k))$$

where L is the number of quantised intervals for the θ, and ϕ are angular co-ordinates. Some histograms representing the point density as a function of the radius ρ are shown in Figures 6.18, 6.19 and 6.20. In Figure 6.21 we can see the projected data volume for one particular solid angle of the actual shoulder data and the resulting surface points. Table 6.1 shows the various effects of modifying the extraction parameters, and Figure 6.22 illustrates it.

As we intend to retrieve the corresponding implicit surface using least-squares optimisation, our data needs to be smooth, meaning sufficiently dense everywhere, and not sporting any holes. The reasons for this will become clear in Section 6.2.3. After a lot of testing, however, we came to the conclusion that no combination yields a result good enough for obtaining such a smooth set of surface points.

At the end of this section, we will display the comparative results obtained with the surface points extracted with the various methods, in order to compare their (un)worthiness.

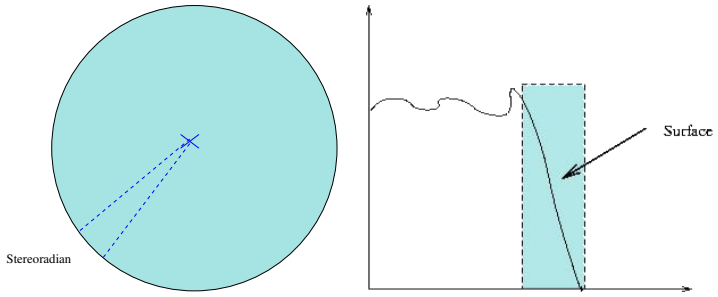


Figure 6.18: (a) Dense data volume. (b) Histogram of the point density as a function of the distance for a solid angle containing dense data. The surface points are taken to be those in the box.

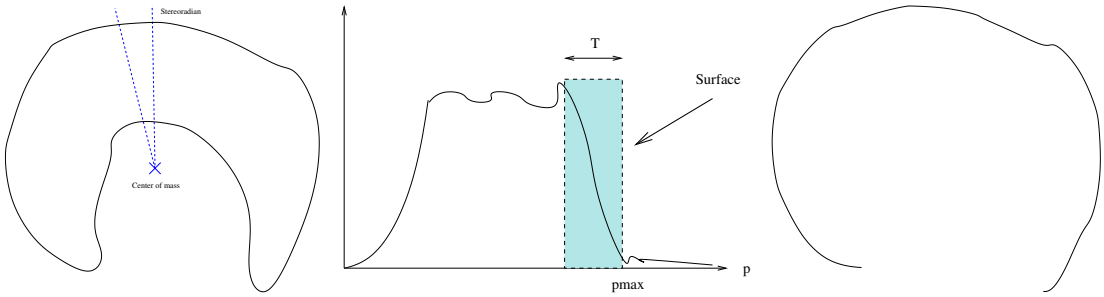


Figure 6.19: (a) Example of a non-convex shape. (b) Corresponding histogram, where one can see one great increase in point density, and one great decrease. However, only the decreasing portion is taken into account for surface extraction. (c) Resulting partially truncated extracted contour.

6.2.1.2 Data slices

The solid angle idea having disappointed our expectations, our next attempt at surface point extraction was to enhance the solid angle method in order to get the best of both worlds, i.e. sufficient detail in surface curvature, as

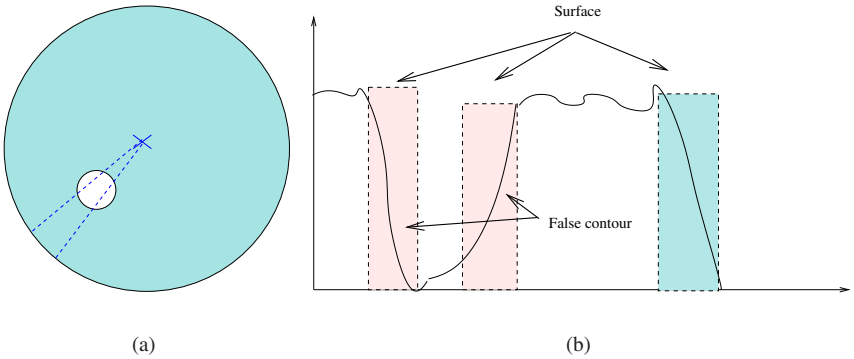
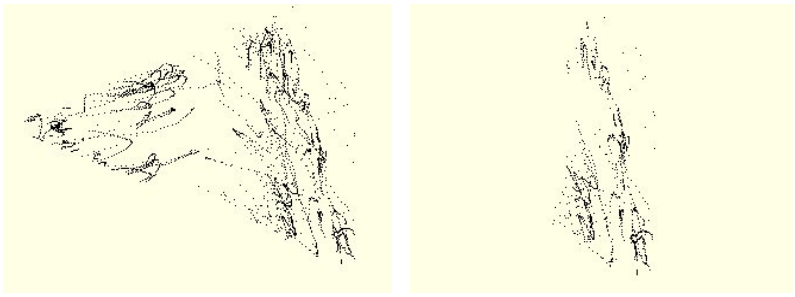


Figure 6.20: (a) Shape containing one hole. (b) Corresponding histogram where the contours of the hole, i.e. the points inside the two boxes on the left, are ignored and only the outer surface is detected.



(a) (b)

Figure 6.21: Surface extraction using real data: (a) data points in one of 49 solid angles used to cover the space of orientations; (b) corresponding surface points, extracted using a distance threshold $T = 0.1$.

Fine-tuning	Effect
Threshold interval large	Points will not be a thin layer
Threshold interval narrow	Only most distant points
Angles too big	Concavities are missed
Angles too small	Points are not continuous and smooth enough

Table 6.1: Fine-tuning of solid angle point extraction parameters.

well as a thin layer of surface points. If we combine a narrow threshold with a large sampling of solid angles, as before, we obtain surface points that are globally insufficiently smooth in order to be fitted to an implicit surface. However, if we slice the surface points obtained by solid angles and then flatten all the points, we obtain a set of points representing the 2D contour of one data slice, as illustrated in Figure 6.24.

From here on, it should be simple enough to extract the continuous contour of these 2D points using a method such as Snakes [Kass *et al.*1987] whose principle is shown in Figure 6.25. Once we have obtained the snake corresponding to each data slice, we could interpolate linearly between the control points of the snake in order to get the intermediate contours and thus construct the surface points representing our initial shoulder compound data. Reconstructing the surface from slices, however, is a complex process and we have therefore not investigated this approach fully.

6.2.1.3 Triangulated meshes

A great variety of methods exists that directly derive from scattered data a boundary representation in the form of a triangulated mesh. Despite the fact that we do not wish to use such a boundary representation for our joint limits, we could nevertheless sample these meshes in order to obtain surface points for our least-squares fitting to an implicit surface. In the following sub-sections, we will explore two such possibilities and draw our conclusions in terms of usability for our purpose.

Shrink-wrapping Shrink-wrapping in computer graphics is directly derived from the industrial process in which film is used to completely and tightly cover a package, it being generally used to apply a thin plastic film over retail type products. The same idea is applied to scattered 3D data, so as to obtain a mesh wrapped as closely as possible around the points, and in our case, the plastic film is represented by a triangulated unit sphere to be shrink-wrapped.

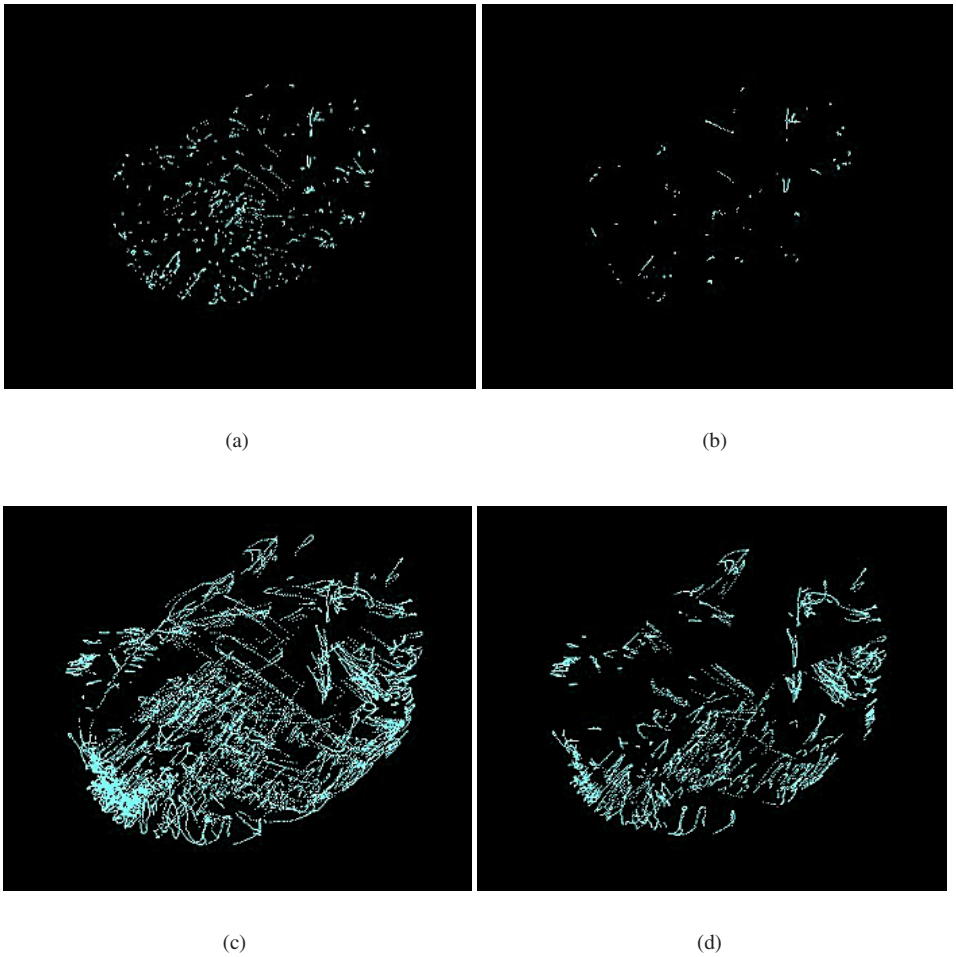


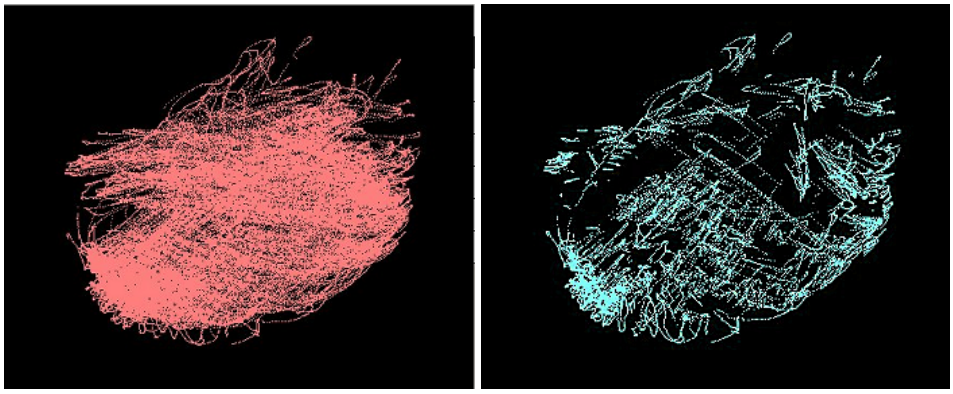
Figure 6.22: Effect of the solid angle extraction parameters on the shoulder compound data: (a) threshold T=0.01 and a resolution r of 20 voxels; (b) T=0.01 and r=7; (c) T=0.1 and r=20; (d) T=0.1 and r=7.

The various steps are illustrated in Figure 6.27 and are the following:

1. The unit sphere containing the data is tesselated using a uniform distribution [Saff and Kuijlaars1997] and the vectors from the centre of the sphere to each tessellation point are derived, as shown in Figure 6.27(a). The tessellation points are Delaunay triangulated, meaning that inside the circumsphere of any triangle there are no other points of this set, as illustrated by Figure 6.26 in 2D. The algorithm used for this triangulation is the one described in [Watson1981]. In this manner, we have created the triangulated mesh that is to be shrink-wrapped.

2. We derive data vectors, which are the vectors from the centre of the sphere to the data points (Figure 6.27(b)).

3. For each data vector, we determine which tessellation vector is the closest by evaluating their dot product, then orthogonally project all the data points onto their closest tessellation vector, as shown in Figure 6.27(c).



(a) (b)

Figure 6.23: From the 3D data points to the surface points: (a) original volume of 3D quaternion data points; (b) extracted surface points.

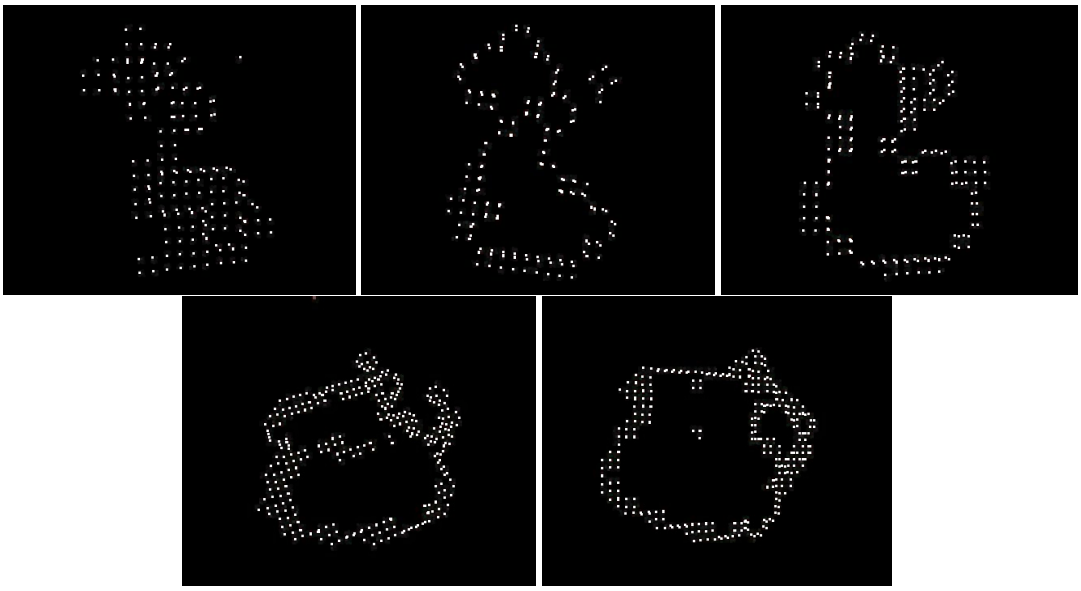


Figure 6.24: Five slices out of 20 of the shoulder compound data surface points extracted using solid angles.

4. The triangulated mesh vertices are translated to the nearest projected data point on their corresponding tessellation vector, thus yielding the shrink-wrapping triangulated mesh (Figure 6.27(d)).

Applying this technique to our shoulder compound data points yields the results shown in Figure 6.28. A tessellation of respectively 500 and 750 points on the sphere produces an insufficient point density, where implicit surface fitting is bound to fail as primitives will grow through the holes. tessellation with 1000 points, as shown in (c) and (d) results in perfectly extracted surface points that are however not usable for fitting as once again the smoothness criterion is not fulfilled.

Furthermore, a too small tessellation ratio will result in a smoothed out surface, as in Figure 6.27(e), where the surface details have been missed. On the other hand, if the tessellation ratio is high, the extracted triangulation

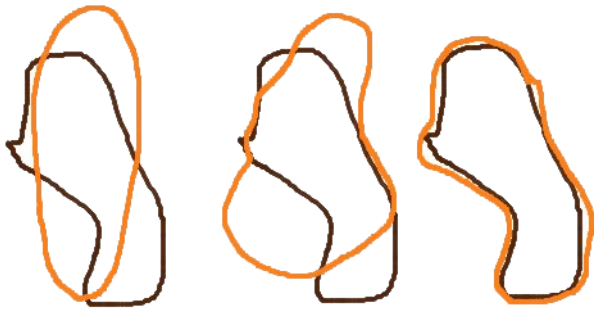


Figure 6.25: The Snakes algorithm: in black, the contour to be detected and in grey, the evolving snake.

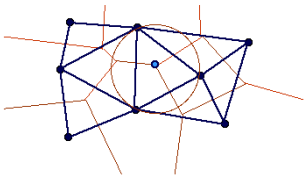


Figure 6.26: A 2D Delaunay triangle and its circumcircle.

may become very spiky in areas where the data is not dense, which is our case. The more uncomfortable zones of the shoulder compound's motion range are sparsely sampled, and in these areas, shrink-wrapping produces spikes, making the resulting surface points impossible to fit, as will be shown at the end of this section.

Alpha-shapes An alternative method that is also based on Delaunay triangulation is alpha-shapes, which is very popular with the scientific community namely for retrieving the shapes of molecules in biological applications. The concept was introduced by [Edelsbrunner and Mücke1994] and their intuitive description of an alpha-shape is that it is a generalisation of the convex hull of a point set. α is a positive real number and its corresponding α-shape is a polytope that is neither necessarily convex nor necessarily connected. If $\alpha = \infty$, the α-shape is the convex hull, and as α decreases, the α-shape shrinks and develops cavities.

To illustrate this in practice and in 2D, one starts out with the computation of the Delaunay triangulation of a set of points. At the beginning, the α-shape is equal to the convex hull. α decreases and for the current value of α, all triangles whose circumcircle have a radius greater than α are excluded from the α-shape. This is illustrated by Figure 6.29.

For computing the α-shapes of our shoulder compound data, we used the Alvis α-shape visualiser developed by the authors of the α-shapes paper. The result is stored as a set of α-ranks, each rank corresponding to a certain value of α. The results obtained for various ranks of α are shown in Figure 6.30.

Using the optimal α value of 0.0827 on our data, we get the triangulated mesh and corresponding vertices shown in Figure 6.31. The boundary shown in Figure 6.31(a) was obtained by extracting the boundary simplices of the α-shape, using the packages software provided with Alvis. If the number of resulting vertices is too low, we sample each triangle in order to yield more points on our surface.

Despite the beauty of the method and the good quality of the extracted surface points, we are still confronted with the same old problem of lack of smoothness, and no automatic fitting of an implicit surface can be performed without involving a lot of testing with various parameter values (see end of this section). As said before, we do not wish to obtain a method for which the parameters' values are case-dependent.

Recently, a new paradigm for designing smooth surfaces was introduced by [Edelsbrunner1999], based on

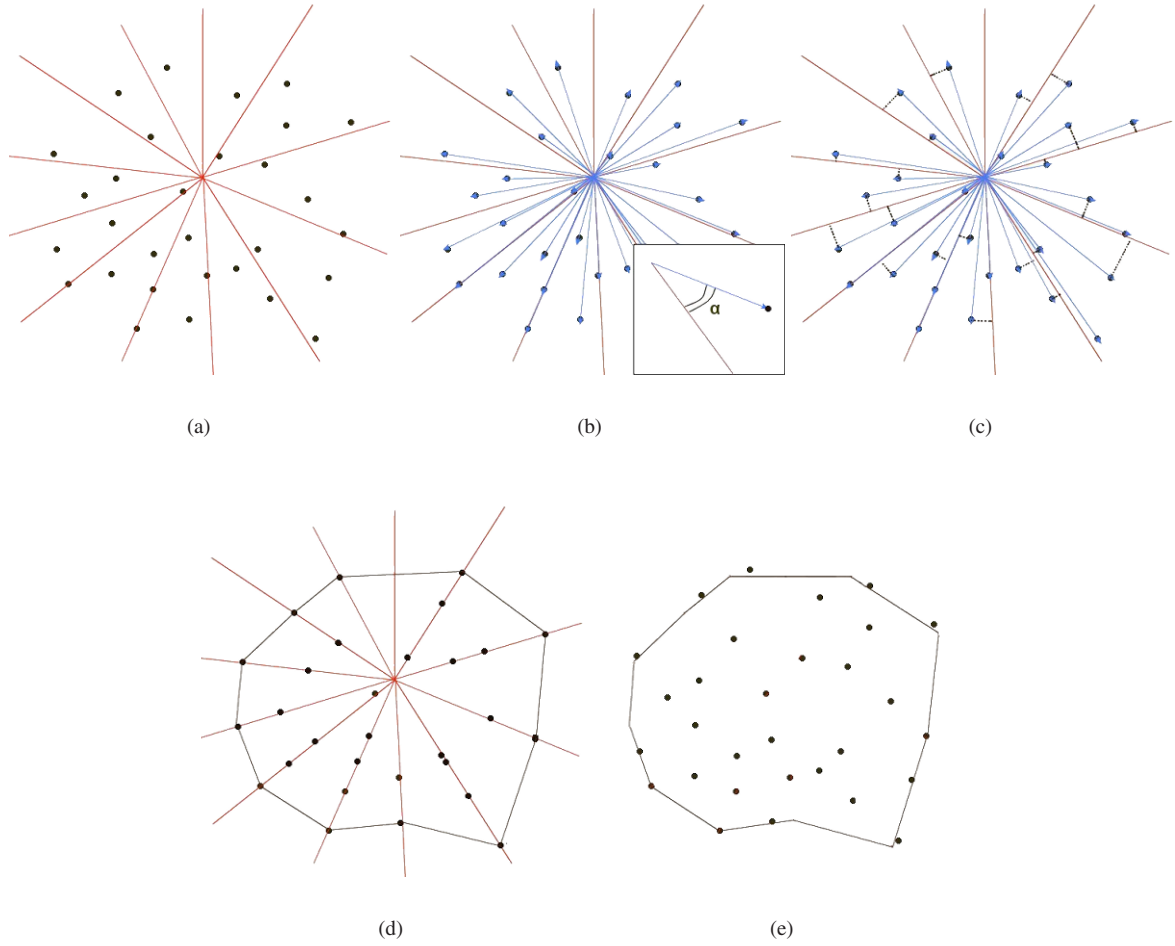


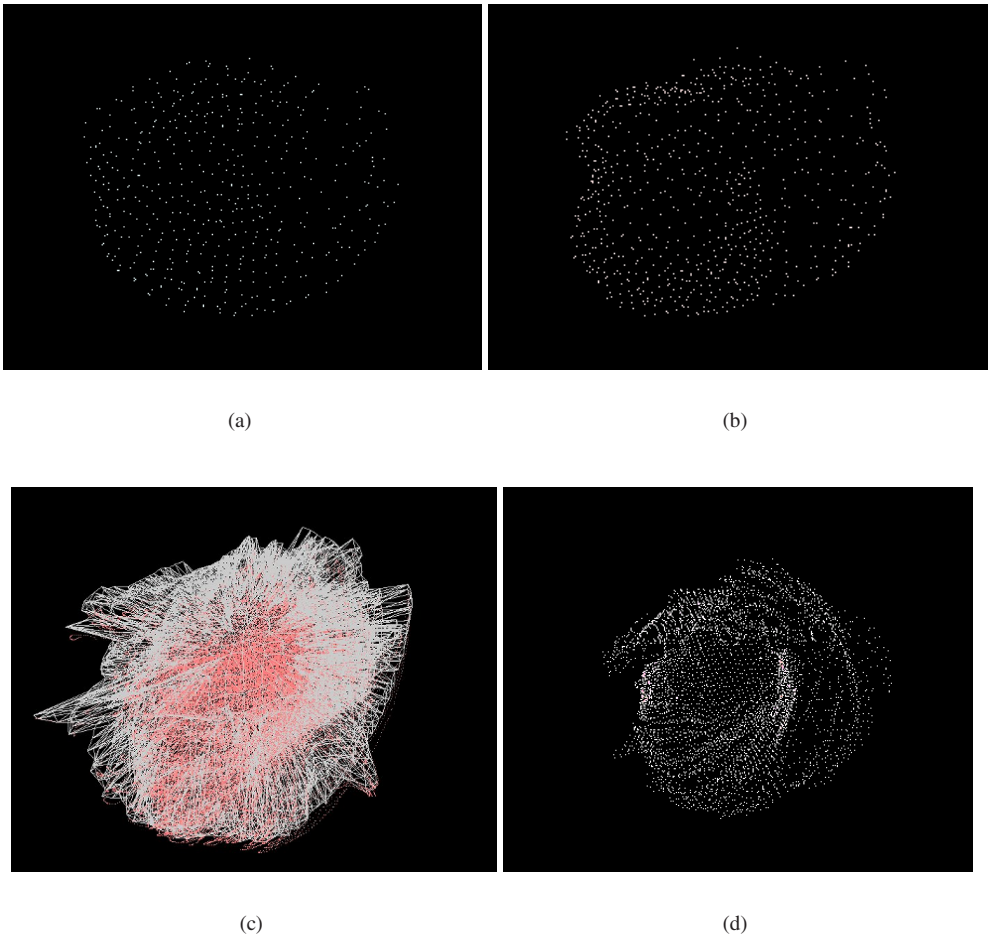
Figure 6.27: Shrink-wrapping example in 2D: (a) initial data with tessellation vectors; (b) data vectors; (c) projection onto tessellation vectors; (d) resulting shrink-wrap; (e) superposition of the initial data and the boundary.

Delaunay, Voronoi and alpha-complexes of a set of weighted points. This new paradigm is called a "skin" and is basically the envelope of a set of spheres, that are real if the weight is positive and imaginary if the weight is negative. However, at the present moment, no software or algorithm are available for rendering such surfaces.

6.2.1.4 Marching Cubes

To visualise the implicit surface as in Figures 6.44(c) or 6.45(b), we are currently using an implementation of the Marching Cubes algorithm [Lorensen and Cline1987], this version having been created in the context of research work on physically-based deformations [Aubel2002]. The Marching Cubes algorithm determines the surface mesh vertices and their normals on the basis of some sort of definition of the surface to reconstruct. Despite the fact that in our case, the surface is not defined, we still have at our disposal some kind of information concerning it: the volumetric data.

Marching Cubes on surfaces In order to modify the Marching Cubes algorithm to fit our needs, we would like to remind the reader of the fundamental principles of the method. In the 3D case, the bounding box is determined



(a) (b)

(c) (d)

Figure 6.28: Surface point extraction using shrink-wrapping for the shoulder compound data: (a) extracted points using a tessellation of 500 points; (b) ditto with a tessellation of 750 points; (c) shrink-wrapping and triangulation with 1000 tessellation points; (d) surface points with a tessellation of 1000 points.

and voxelised, i.e. divided into a grid of cubes, with each cube being assigned a position *(i, j, k)* in the grid. In this implementation, a ray is cast at incremental angles to determine the "seed voxel", i.e. a first voxel belonging inside the surface. The neighbouring voxels are then explored in turn, meaning that this method will only be able to reconstruct one object, and not more. However, in our case, we do not expect to have discontinued joint limits, so we didn't deem it necessary to extend it to multiple objects, For each vertex of each neighbouring voxel it is determined whether the vertex is inside or outside the surface. An edge of a voxel intersects the surface if one of its vertices is inside and the other outside. For each voxel, there are 23 possible ways in which a surface can intersect its edges (see Figure 6.32) and for each browsed voxel, the corresponding configuration identifier is stored. When the surface is to be rendered, each voxel is drawn with its corresponding triangles and normals, thus considerably speeding up the rendering process.

We here recall the expression of the field function of our spherical primitives:

$$f_i(P) = \begin{cases} -k_i r + k_i e_i + 1 & \text{if } r \in [0, e_i] \\ \frac{1}{4}\left[k_i(r - e_i) - 2\right]^2 & \text{if } r \in [e_i, R_i] \\ 0 & \text{elsewhere} \end{cases} \tag{6.6}$$

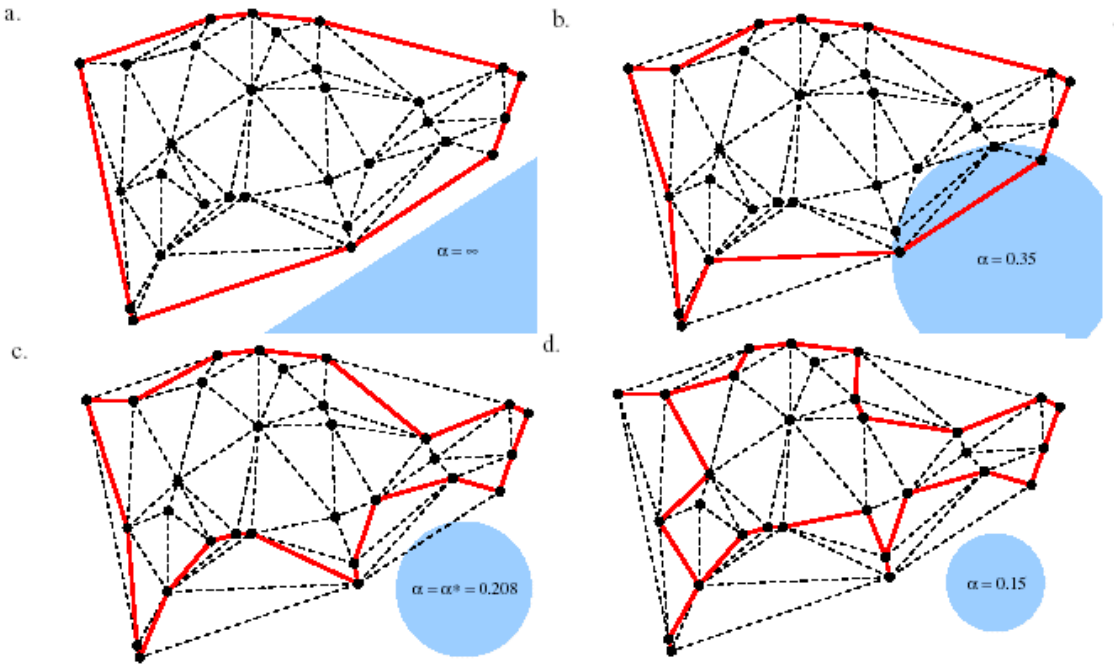


Figure 6.29: The α-shapes for decreasing values of α.

where $R_i = e_i + \frac{2}{k_i}$. The iso-surface derived from such a set of primitives is defined as:

$$S = \{\ P(x,y,z) \in \Re^3 \mid F(P) = iso\ \}$$

$$F(P) = \sum_{i=1}^{n} f_i(P)$$

For rendering such an implicit surface, the density with respect to the iso-surface function is evaluated for each voxel vertex. If this sum of field functions is smaller than the surface iso-value, then the point is outside the surface and a value of 1, for example, is returned. For a point inside the surface, the value could typically be -1.

Marching Cubes on voxelisation In our case, we have at hand only the cloud of data, but this nevertheless gives us information on the volume that should be covered by our surface. Therefore, the idea is to perform a voxelisation of our data, and then on top of this, run the Marching Cubes algorithm to extract the surface. If necessary, the result can be cleaned up *a posteriori* using erosion and dilation-like techniques.

Data voxelisation To perform voxelisation on our data, we first determine its bounding box and the latter is then divided into a regular grid of $m \times m$ voxels of width w each, these depending on the chosen resolution. We then compute the number of points contained within the cube of width w around the centre of mass and normalise this number with respect to voxel width, this becoming the reference density.

For each voxel, the density is computed. From here on, we can either set a threshold that determines which voxels are to be considered inside the volume or not and stop there ("simple voxelisation"), or we can recursively sub-divide any voxel whose density is smaller than the reference density. The danger of the recursive sub-division option is that around the borders of the volume, density is lower and we might end up with a multitude of extremely small voxels containing a single point. This fact does not cause any problems as such, except that it drastically increases the number of voxels belonging to the volume.

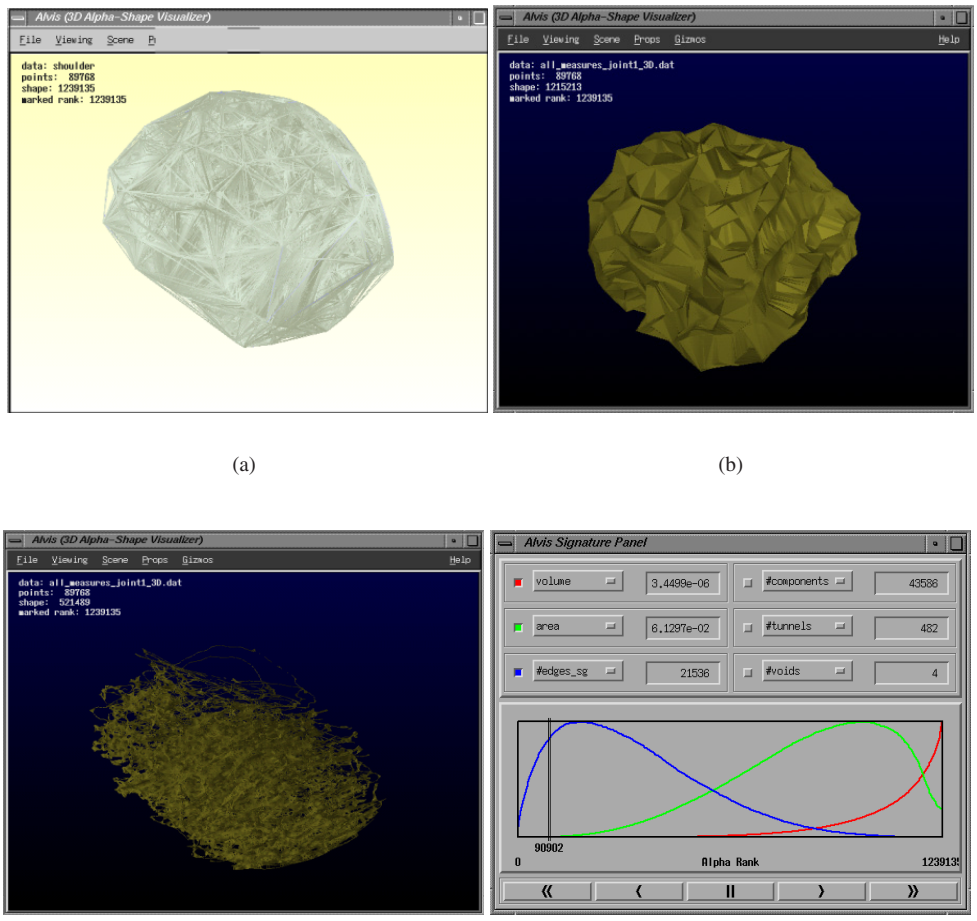
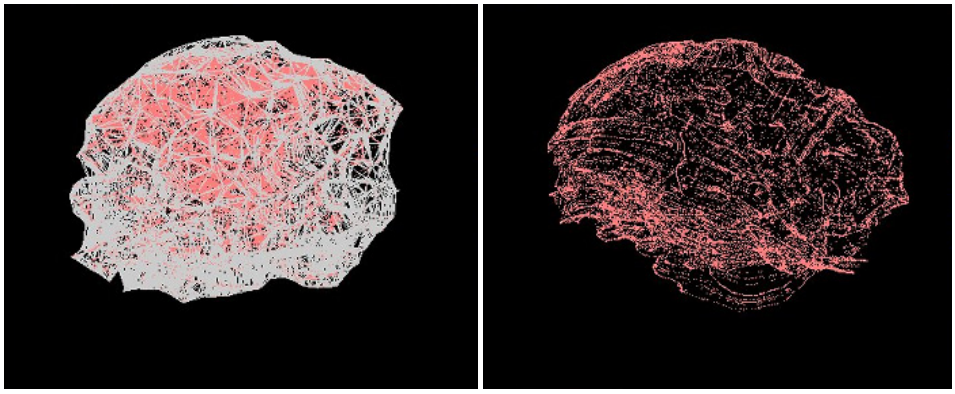


Figure 6.30: Shoulder compound α-shapes in Alvis: (a) α has maximum value and is the convex hull of the data set; (b) α=0.0827 for α-rank 1'215'213 is the optimal α value as shown in (e) and is the retained α-shape for our data set; (c) α decreases to rank 521'489, cavities start to appear in the surface, disconnecting the triangles; (d) the Alvis panel displaying the interval of α-ranks, from 0 to 1'238'135.

Whichever voxelisation scheme is chosen, at the end of day (NB: this is just an expression and not an indication of actual computation time), we have an array of voxels that represent our volume. To avoid any confusion, we will from here on refer to "voxels" as the voxels resulting from the sub-division of the n-dimensional space, and to "positive voxels" as those that are effectively triangulated by Marching Cubes. The resulting voxelisation of the shoulder compound data is shown in Figure 6.33.

Marching Cubes In order to apply the Marching Cubes algorithm to the previously-mentioned voxelisation, we need to define, for a voxel, two essential functions.

The first one is the computation of the intersection of a voxel with a ray, in order to determine the seed voxel. For this, we used *Graphics Gems* code for fast ray-box intersection [Woo1990]. The second function is the one that returns a value stating that a given point is either inside or outside a voxel. Determining whether a point is inside a voxel or not is trivial, and -1 or 1 is returned respectively for each case.



(a) (b)

Figure 6.31: Shoulder compound α-shapes for α=0.0827: (a) the triangulated mesh and (b) the extracted surface points.

We are now able to run Marching Cubes on our voxelisation, and we obtain the result shown in Figure 6.34.

It is not clearly visible in Figure 6.34(b), but rotating the cloud of points in 3D reveals data points on the interior of the cloud, not only on the boundaries. This is due to the fact that areas of low density resulted in empty voxels, and therefore neighbouring positive voxels are considered to be on the surface.

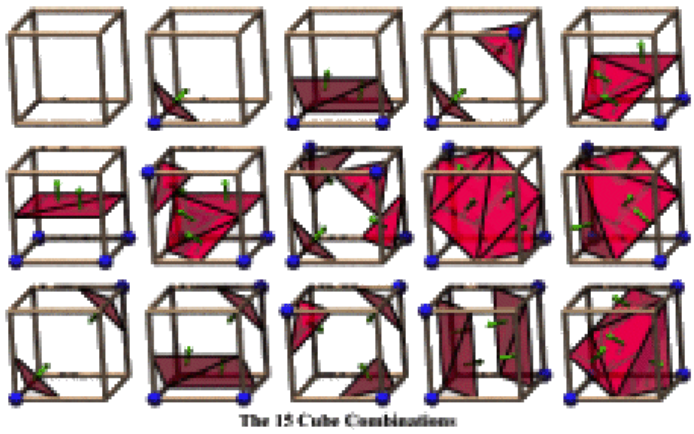
Dilation In order to try and overcome the previous setback, our idea is to try and fill the areas of lesser density, i.e. perform a dilation operation, but only on the interior of the volumetric cloud of data. For this, we will consider every voxel to be 26-connected[2].Each voxel that is empty but has a large number of positive voxels as neighbours will be added to the array of positive voxels. The following iterative process is applied for a bounding box of width n voxels:

```
for each voxel (i, j, k)
 if the voxel is positive then
   if the voxel has i, j or k equal to 0 or n then
    the voxel belongs to the boundary
   end if
   if the voxel has a low number of positive neighbours then
    the voxel belongs to the boundary
   end if
 else if the voxel has a high number of positive neighbours then
  the voxel belongs to the interior
  add the voxel to the list of positive voxels
 end if
end for
```

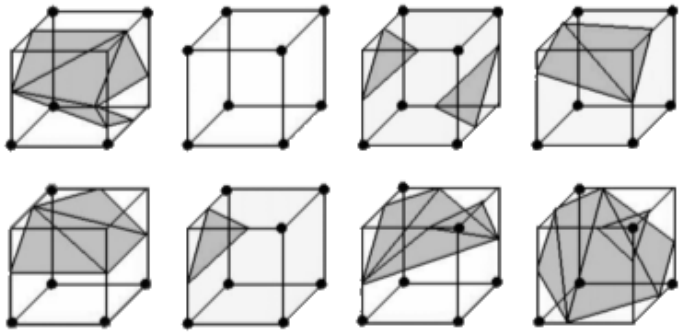
Once all positive voxels are thus classified as either boundary or interior, we parse the array once more and sample the positive voxels, thus generating a large amount of data points within all interior positive voxels. In Figure 6.35, we show in (a) the interior positive voxels detected by exploring positive-neighbour-connectivity. In (b) is depicted the resulting augmented cloud of data.

We now apply Marching Cubes once again and we indeed no longer obtain surface points on the interior, but only the effective boundary of our initial volume, as shown in Figure 6.35(c).

[2]A voxel with a connectivity of 26 neighbours means that this voxel has 26 direct neighbours.



(a)



(b)

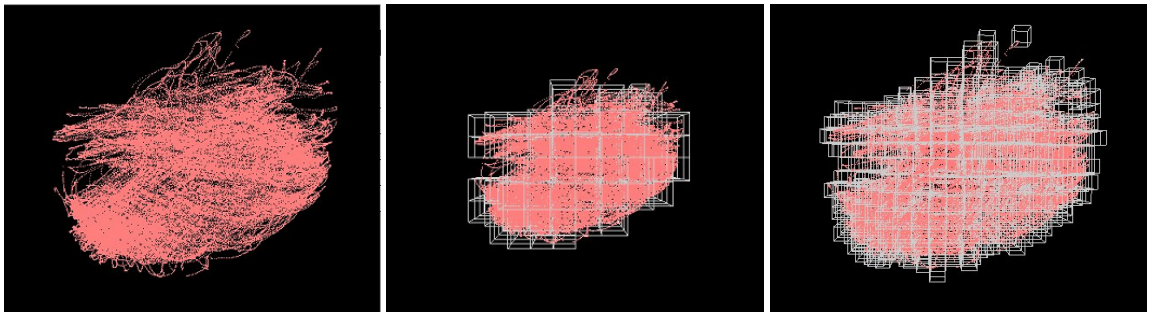
Figure 6.32: Marching Cubes: (a) the 15 ways to triangulate a voxel, as per Lorensen's initial algorithm; (b) the 8 extra cases added by [Shoeb1998] as for a same configuration of edge intersections, there is actually more than one way to triangulate the voxel. Not taking these ambiguities into account can lead to holes in the displayed surface.

We have presented the various techniques we applied to our quaternion data in the hope of extracting surface points from it. None of them delivered sufficiently in this respect, the fault residing with our data, whose varying density presents an unsurmountable obstacle to traditional surface point extraction methods.

6.2.2 Primitive-per-voxelisation approximation

Attempting to fit an implicit surface to surface points extracted using various techniques does not yield satisfactory results. In order for any of the methods to function, it would require not only for the data points to have a relatively smooth shape, but also for them to be sufficiently dense. In the absence of these criteria being met, the implicit surface fitting performance will always be average.

Having come to the conclusion that our data is neither smooth nor dense enough to ever hope to extract a surface from it, we propose a solution that directly deals with the volumetric data, the idea being inspired by



(a) (b) (c)

Figure 6.33: Voxelisation of shoulder compound data, where only positive voxels are shown: (a) 3D data cloud; (b) simple voxelisation with a resolution of 10 x 10 and a threshold of 1% of the reference density; (c) recursive voxelisation with the same resolution.

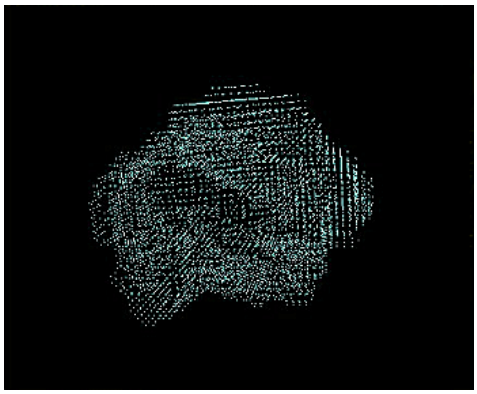


Figure 6.34: Marching Cubes on the voxelisation of shoulder compound data: vertices of the reconstructed mesh.

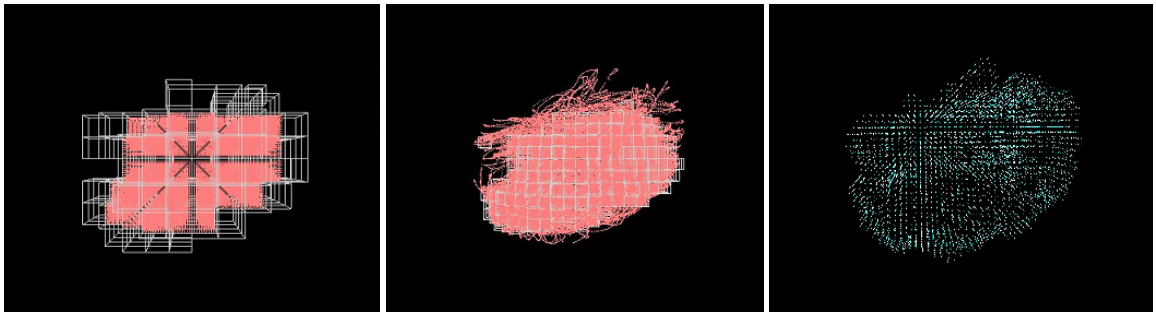
the surface point extraction technique of Section 6.2.1.4. Leaving out the step that is causing the headaches, we directly proceed from voxelisation to implicit surface. As our array of positive voxels already represents our shape and as we are able to fill the gaps resulting from less dense sampling, we could simply fill each positive voxel with a spherical primitive and then obtain the iso-surface from this cluster of primitives.

As shown in Figure 6.36, for every positive voxel obtained from our cloud of data, we place in the middle of it a spherical primitive whose thickness e_i is determined by eq.(6.6), given that the radius of influence R_i is half of the width of the voxel[3]. The value of the stiffness k_i should not be too high, as otherwise the overall set of primitives will grow too large, but neither should it be too small as otherwise the primitives will not blend enough, in which case we might obtain holes within the volume. By our experience, the best choice for the stiffness k_i is more or less 20, for a voxelisation resolution ranging from $10 \times 10 \times 10$ to $20 \times 20 \times 20$ voxels.

Voxelising our shoulder compound data, and then filling the positive voxels yields the iso-surface shown in Figure 6.37(b). What iso-value to choose depends on how closely we want our contour to cling to our spherical primitives. A too high value will yield disconnected spheres, as shown in Figure 6.37(c), and a too low value will create a too distant contour. We have found that the ideal interval in all cases is $[5.0, 7.0]$.

To see how closely our envelope fits our data, we can display the implicit surface in wire-frame, as in Figure

[3]The image depicts purely the principle of placing the spherical primitives within the voxels. It is understood that no holes will be present in the resulting surface.



(a) (b) (c)

Figure 6.35: Dilation on interior voxels: (a) Sampled interior positive voxels detected for the data points of Figure 6.34(a); (b) augmented 3D data, including data from positive boundary voxels; (c) Surface points of shoulder compound data extracted by gap-filling and Marching Cubes.

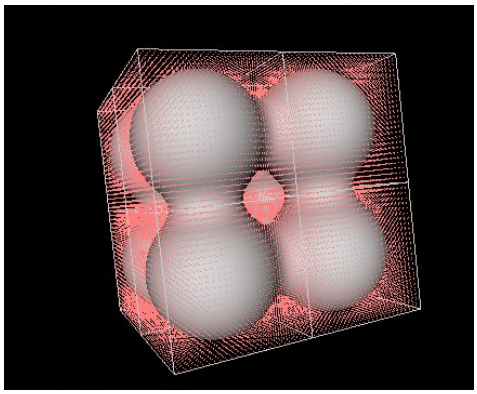


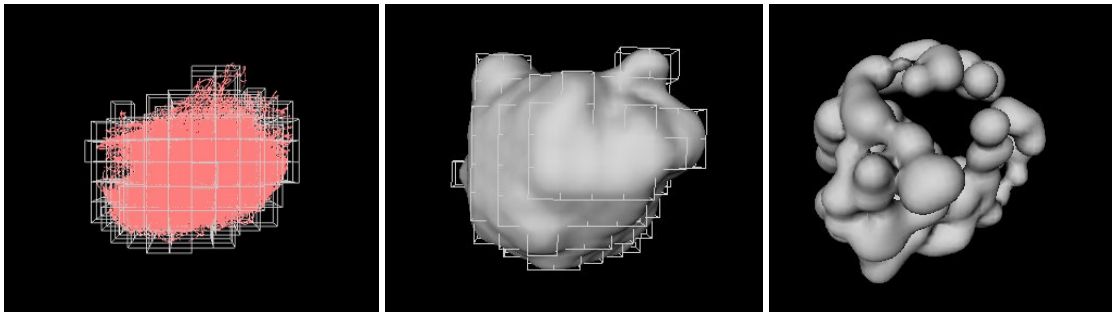
Figure 6.36: Filling a voxelisation with spherical primitives.

6.38(a) for an iso-value of 5.0. In the latter case, the contour does not cling too closely to the data, but on the other hand, it does enclose the sparse data towards the surface of the data cloud. If we choose an iso-value of 7.0, the envelope will be tighter, but will then not contain such scattered data. The scattered data in question is not noise but valid shoulder rotations, so it should not be discarded as its presence is due to under-sampling during the motion capture session.

Once our implicit surface obtained, we can render it using the Marching Cubes algorithm and a high-resolution, in order to yield a very dense set of surface points, as shown in Figure 6.38(b).

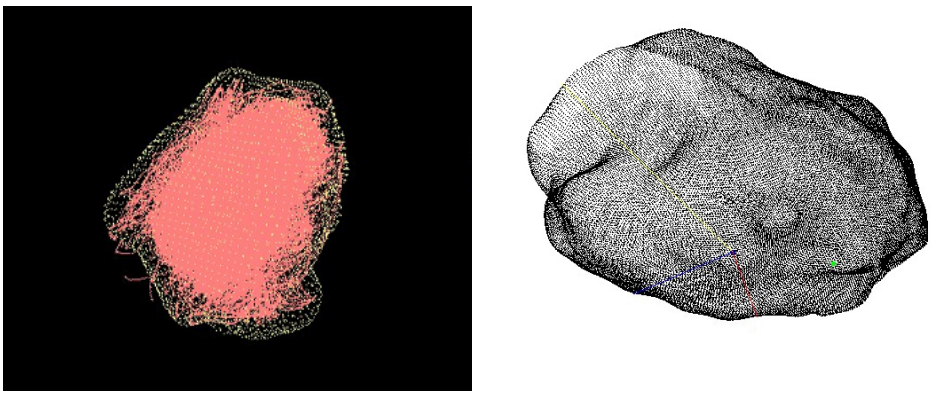
6.2.3 Implicit surface complexity reduction

For enforcing joint limit constraint, we can very well use the implicit surface computed on the basis of the spherical primitives placed within each voxel, but a "lighter" version of the implicit surface would be desirable. Indeed, given that we presently have one spherical primitive at each voxel location, if the grid resolution is high, the number of primitives can range from several hundreds to a good thousand, which will become costly in terms of computation time. It is therefore in our interest to find a reduced representation for the same surface. By generating the mesh corresponding to the derived implicit surface, we can obtain a set of surface points as dense as we like. Using these points, we can then approximate them by an implicit surface, by placing a first primitive within the cloud of surface points, and then recursively sub-dividing it and optimising its parameters until the



(a) (b) (c)

Figure 6.37: Implicit surface for shoulder compound data: (a) simple 10×10 voxelisation; (b) extracted iso-surface with an iso-value of 5.0; (c) iso-surface for an iso-value of 10.0 results in disconnected spherical primitives.



(a) (b)

Figure 6.38: Superposing the data and the contour: (a) the original data and its enveloping implicit surface; (b) high-resolution mesh vertices.

resulting implicit surface fits the surface points as closely as possible.

The idea of using implicit surfaces for the purpose of approximating scattered data was first reported in [Muraki1991]. His algorithm positions a skeleton at the centre of mass of the points and sub-divides it until the process reaches a correct approximation level. This, however, has several drawbacks, such as the lack of a local criterion and a high computational cost. Furthermore, the algorithm does not perform well on data with "holes", and requires knowledge of surface normals.

This early work was later extended [Tsingos *et al.*1995] by introducing a more efficient way to split the implicit surfaces without requiring normals and expressing them in terms of locally defined field functions as in eq.(6.6). This allows the use of an iterative method based on minimising the distance between the real points and the generated surface.

Using the defined field function, we are left with five parameters to optimise for each primitive, for a given iso-value: e_i, k_i, x_i, y_i, z_i, where (x_i, y_i, z_i) are the 3D co-ordinates of the primitive's centre.

To compute the implicit surface S so that it approximates the data as well as possible, we look for a set of primitives that minimises in the least-squares sense the energy E

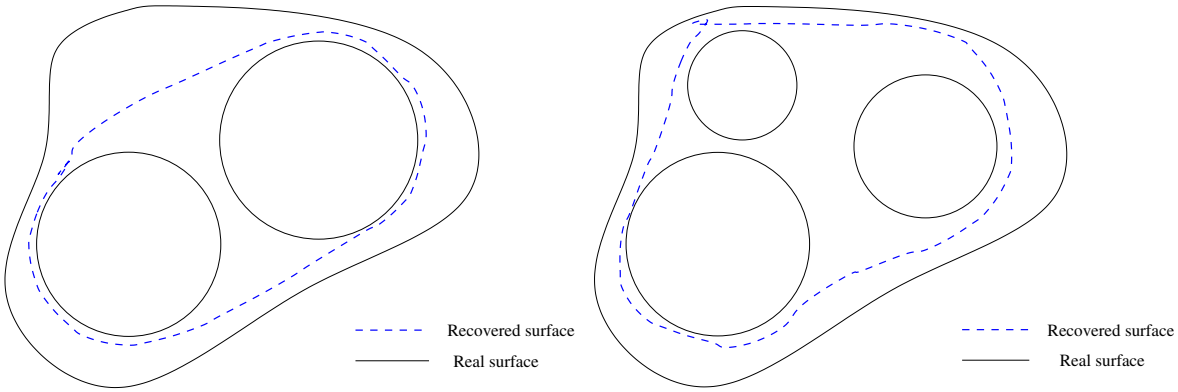


Figure 6.39: Splitting procedure. The implicit surface with the worst fit to the surface points, i.e. the largest C_i as defined by eq.(6.8), is divided into two new ones that are then re-optimised.

$$\begin{array}{l} E = \frac{1}{N}\left(\sum_{j=1}^{N}(f(P_j) - iso)^2\right) + \frac{1}{M}\left(\alpha_1 \sum_{i=1}^{M} v^{+\beta e_i}\right) \\ +\frac{1}{M}\left(\alpha_2 \sum_{i=1}^{M}\left[(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2\right]\right) \end{array} \tag{6.7}$$

where M is the number of implicit surface primitives and N is the number of 3D points. The first term of E forces the surface to approximate the points. The other two terms respectively prevent the surface primitives from wildly increasing in size and from moving too far away from the centre of mass. Without these terms, we would obtain large tangent surfaces. The values of α_1, α_2 and β can be fine-tuned depending on how close we already are to the data points. In general their value is between 0.0 (no constraint) and 0.2 (high constraint). For details about the least-squares fitting procedure, see the Appendix. The gradients for each parameter are derived directly from eq.(6.6):

$$\frac{\partial f_i(P)}{\partial R_i} = \begin{cases} -k_i & \text{if } r \in [0, e_i] \\ \frac{k_i^2}{2}(r - e_i) - k_i & \text{if } r \in [e_i, R_i] \\ 0 & \text{elsewhere} \end{cases}$$

$$\frac{\partial f_i(P)}{\partial x_i} = \frac{\partial f_i(P)}{\partial R_i}.\frac{(x - x_i)}{r}\,, \quad \frac{\partial f_i(P)}{\partial y_i} = \frac{\partial f_i(P)}{\partial R_i}.\frac{(y - y_i)}{r}\,, \quad \frac{\partial f_i(P)}{\partial z_i} = \frac{\partial f_i(P)}{\partial R_i}.\frac{(z - z_i)}{r}$$

$$\frac{\partial f_i(P)}{\partial e_i} = -\frac{\partial f_i(P)}{\partial R_i}$$

$$\frac{\partial f_i(P)}{\partial k_i} = \begin{cases} e_i - r & \text{if } r \in [0, e_i] \\ (r - e_i)\left[\frac{k_i}{2}(r - e_i) - 1\right] & \text{if } r \in [e_i, R_i] \\ 0 & \text{elsewhere} \end{cases}$$

As our primitives should not have negative radii, we have added a penalty function, or "limit spring constraint" to the thickness e_i [Badler *et al.*1993]. We want R_i to be greater than zero, meaning that $e_i - \frac{2}{k_i} > 0$. Hence, if the value of e_i becomes smaller than $-\frac{2}{k_i}$, then a very large value of energy E is returned to prevent further optimising of the parameter e_i in this direction.

To find the optimal set of primitives, we implemented the following automated sub-division scheme. We first place one spherical primitive at the centre of mass defined by

$$x_c = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \quad y_c = \frac{1}{N}\sum_{i=1}^{N} y_i\,, \quad z_c = \frac{1}{N}\sum_{i=1}^{N} z_i$$

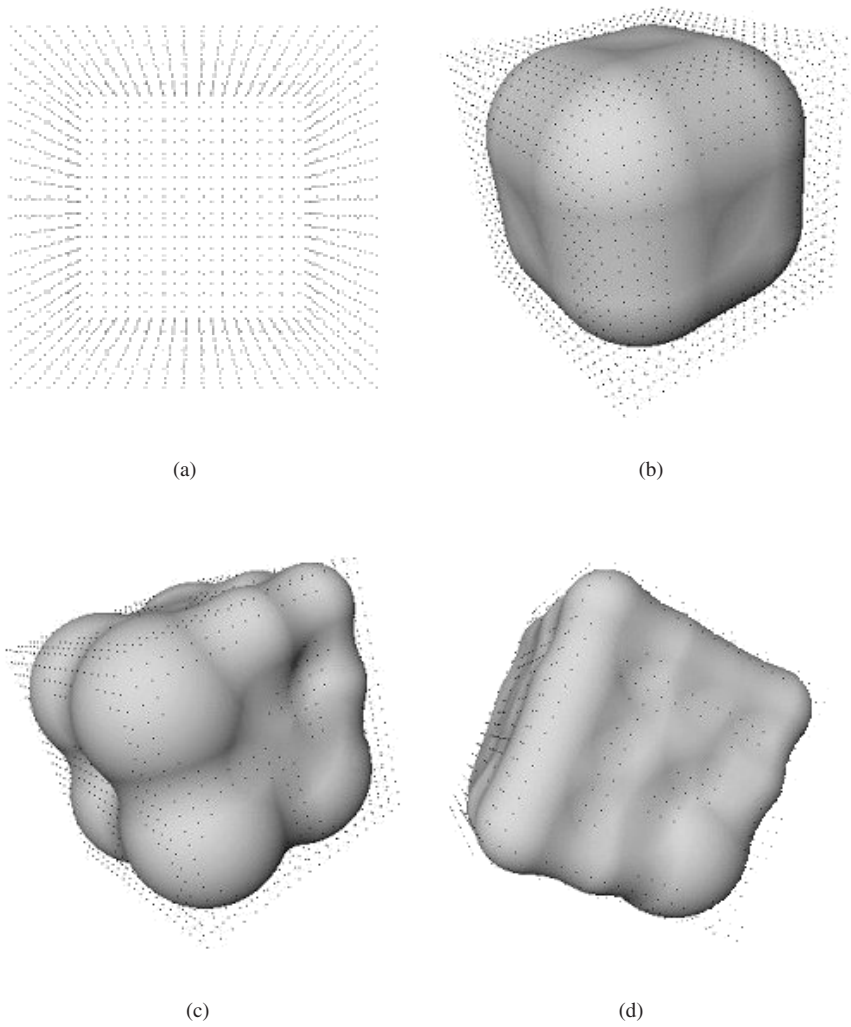


Figure 6.40: Reconstruction of the surface of a cube: (a) surface points; (b) to (d) reconstruction with a division-by-8 scheme.

where N is the total number of data points, and x_i ,y_i, z_i are the co-ordinates of the ith 3D point. The v parameter is the variance of the cloud defined as:

$$v = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2}$$

We then recursively split the primitives based on a criterion C_i computed by summing the contributions of the m_i points that are inside the sphere of influence of the primitive S_i:

$$C_i = \frac{1}{m_i}\left(\sum_{j=1}^{m_i}(f(P_j) - iso)^2\right) \tag{6.8}$$

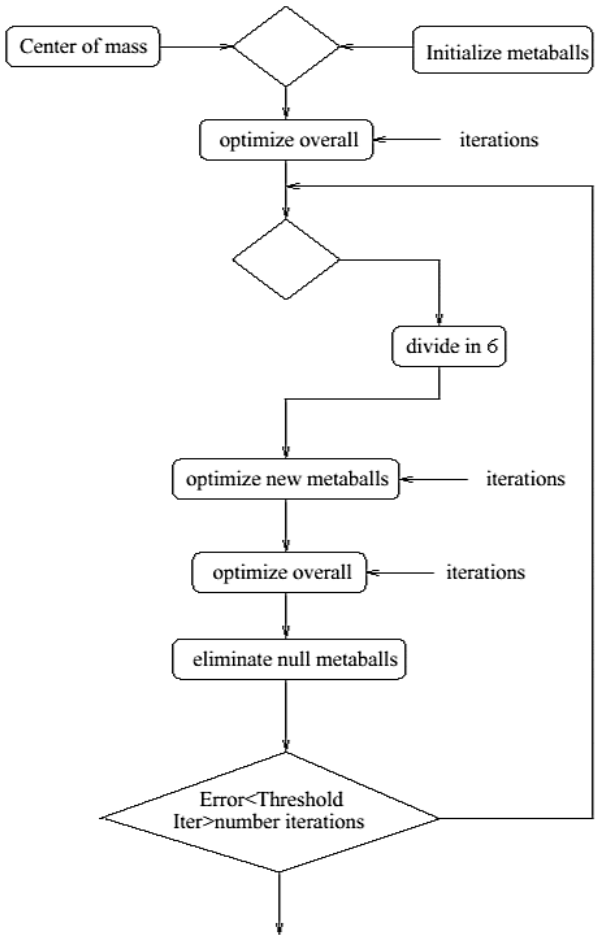


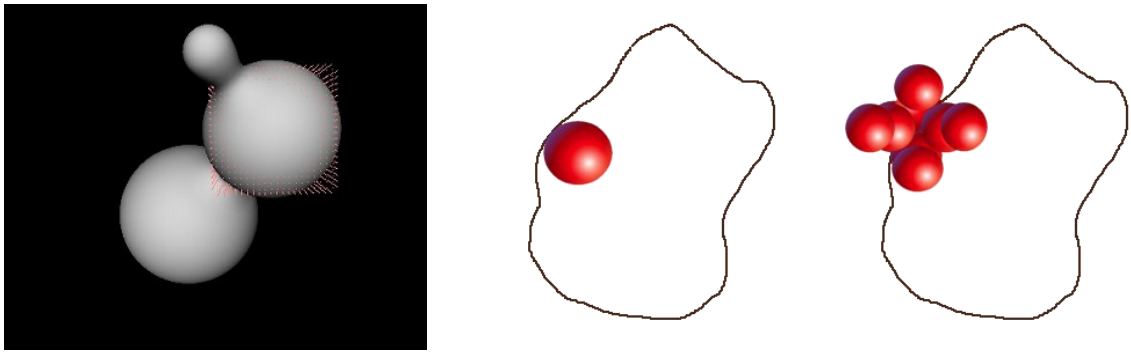
Figure 6.41: Fitting algorithm with primitive division-by-6.

As shown in Figure 6.39, the surface primitive with the largest C_i is sub-divided first. This makes sense because this primitive is the one with the largest first term in eq.(6.7) and, therefore, is the one whose area of influence corresponds to the zone of the surface where the reconstruction is worst. Each split involves sub-dividing the primitive into n new ones that are positioned in a symmetrical manner, so as to yield a shape as close as possible to the original. We then re-optimise the parameters of all the primitives to minimise the full criterion of eq.(6.7).

Various reconstruction tests have been performed with sub-division into respectively 2, 6 or 8 primitives at each step. Division by 2 logically performs the fastest, but is unable to reconstruct sharper edges, which readily succeeds with a division-by-8 scheme (see Figure 6.40). However, computation time increases drastically, as does the appearing of sporadic primitives outside the data cloud. We finally settled for the trade-off of division-by-6, placing the new primitives symmetrically on one axis. The complete algorithm is shown in Figure 6.41.

If surface point density is too low, the regions of sparsity have a tendency to be considered as gaps, through which primitives can grow and sub-divide, thus yielding an implicit surface that is locally overgrown. This phenomenon is illustrated in Figure 6.42. To alleviate this problem, the following check-ups are performed at various stages:

- when splitting a primitive, check that all new primitives are within the cloud of data. For this, we consider



(a) (b) (c)

Figure 6.42: Spurious primitives: (a) If the α and β parameters are set to zero, there is no control over the division, and tangent surfaces appear. (b) and (c) Illustration of the problem of sub-division with child primitives that end up outside the surface.

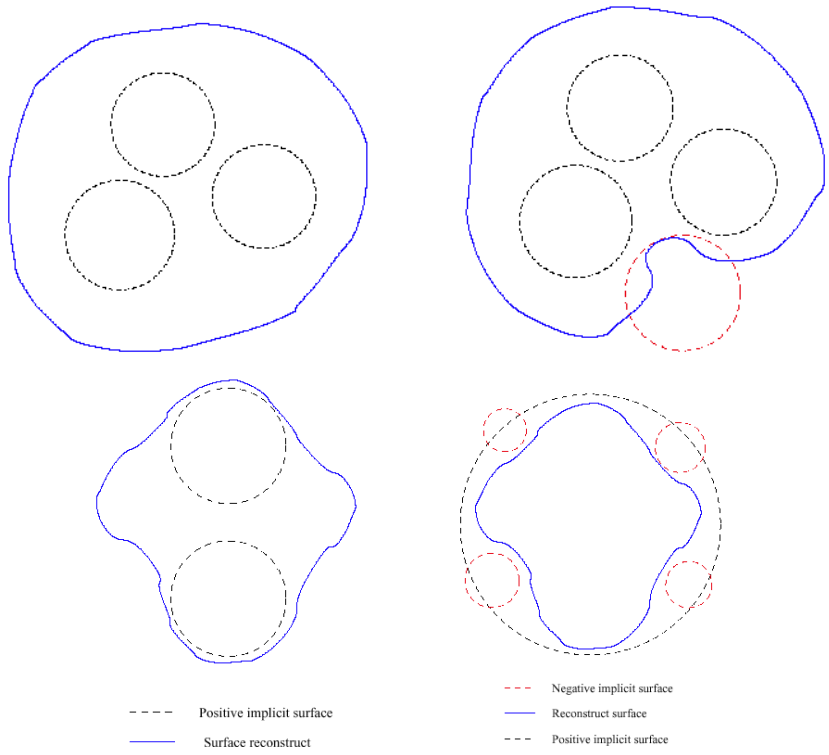


Figure 6.43: Two examples of surface reconstruction with positive and negative primitives.

all data points contained within the solid angle including the new primitive. If a small fraction of these points is closer to the centre of mass of the data than the centre of the new primitive, then the latter is considered to be outside, and is therefore not created (Figure 6.42(b));

- after each least-squares iteration, we remove all spherical primitives whose radius is larger than 1.0, as this

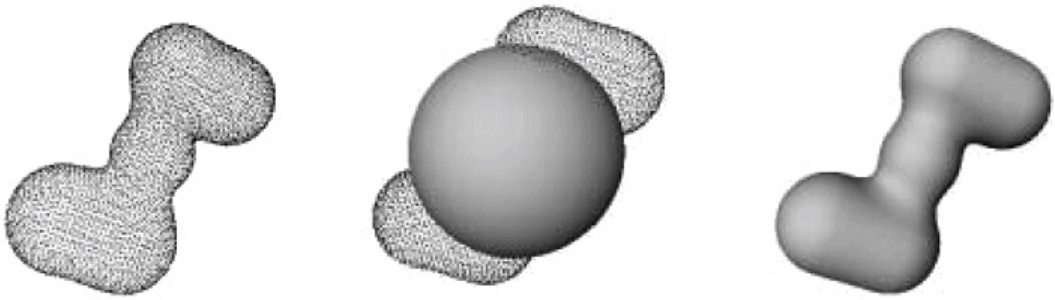
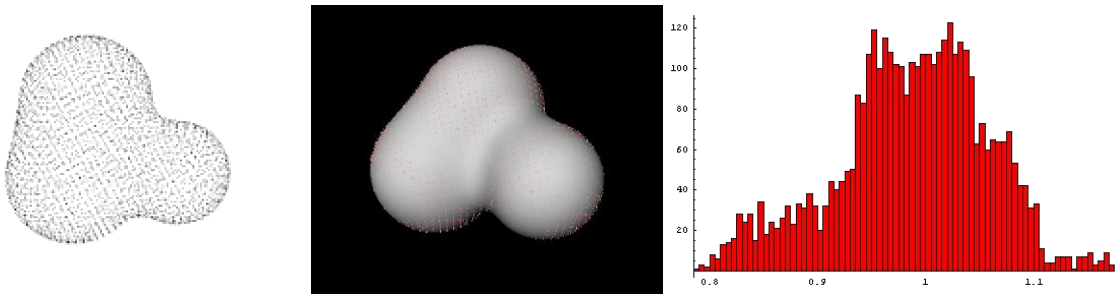


Figure 6.44: Reconstruction of a synthetic object: 1) synthetic 3D point cloud; 2) initialisation of the algorithm with an implicit surface in the centre of mass; 3) final result of the reconstruction with five primitives.



(a) (b) (c)

Figure 6.45: Reconstruction of synthetic data: (a) 3D data cloud; (b) reconstructed shape; (c) histogram of the density values of the data points.

would obviously include the entire cloud of data of unit quaternions;

- after each iteration, we remove the primitives whose centre is further away than 1.0 from the centre of mass of the cloud. As we are using only positive field functions, any primitive at such a distance should not belong to the surface;
- after each iteration, we remove the primitives whose point density is tiny with respect to total density.

After a first least-squares minimisation has been carried out, we again minimise the criterion of eq.(6.7), without sub-division of the primitives this time. This removes the primitives that grow through the holes but may still leave some that simply cover the holes, in concave regions whose shape is difficult to accurately approximate by spherical primitives. This could be addressed by introducing negative implicit surfaces, which can be used to model regions of concavity in the data (Figure 6.43).

The validity of the sub-division algorithm has been tested using synthetic data. The objects were first converted to a cloud of 3D points, and then reconstructed. The results are depicted by Figures 6.44 and 6.45. In Figure 6.45(c), we show a histogram of the value of the energy E of eq.(6.7) for the N points used to derive the shape shown in Figure 6.45(b). Note that the histogram is correctly centred around the chosen iso-value 1.0.

We have designed a least-squares fitting algorithm of an implicit surface to data points, using recursive sub-division of the primitives, minimising a total error function corresponding to the distance from the data points to the implicit surface. Having tested the algorithm on synthetic data, we are now in a position to apply it to our real

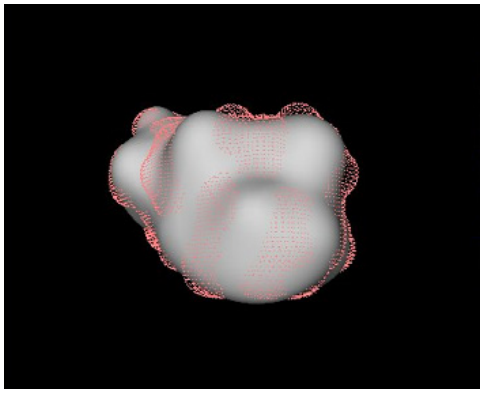


Figure 6.46: Fitting of an implicit surface to the surface points of the initially reconstructed iso-contour.

data with the objective of simplifying the implicit surface joint limits that currently consist of several hundreds or thousands of primitives. The aim of this procedure is to obtain a joint limits representation with as few primitives as possible, as their number will necessarily cause a slow-down in joint limits enforcement.

6.2.3.1 Shoulder compound implicit surface

As the shoulder compound quaternion data served the purpose of illustrating our method, we already have obtained for it the implicit surface corresponding to it, using the primitive-in-voxelisation method. The final result was displayed in Figure 6.37. The only step remaining to be applied is complexity reduction. For the extracted surface points, we have performed least-squares fitting with division-by-6, using four series of four iterations, with α values of value 0.05 to constrain primitive positioning and size. The result can be seen in Figure 6.46, for the surface points extracted using the primitive-in-voxelisation method that were shown in Figure 6.38. We note that there is a small loss of precision as the fit is not absolutely perfect but the gain in processing time makes this trade-off acceptable.

For comparison purposes, we will here present the surface points extracted using the other four methods that we considered too low quality, and show the resulting implicit surface fitting we obtained, which should convince any skeptical reader.

The result in Figure 6.47(a) was obtained the extracted surface points of Figure 6.23(b) that were derived using the solid angles method, with a small threshold and relatively large angles. A fit was therefore possible, as the layer of surface points was thin, but the details of the shape were lost.

For the shrink-wrapping technique, the result is shown in Figure 6.47(b), for the data of Figure 6.28(d).

The points extracted using the alpha-shapes, and shown in Figure 6.31(b), are fitted to an implicit surface using the least-squares optimiser, and yield the result in Figure 6.47(c).

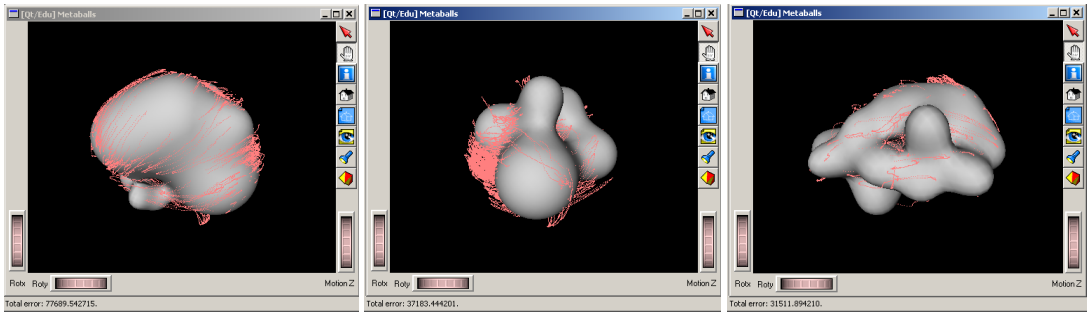
For the method of Marching Cubes surface extraction, the result is displayed in Figure 6.47(d).

The error displayed in the status bar corresponds to the total point density. The higher the density, the more points are contained within the implicit surface. Figure 6.48 shows the superposition of the iso-surface wire-frame with the original volumetric data points, where the closeness of each respective fit to the points can be visually assessed.

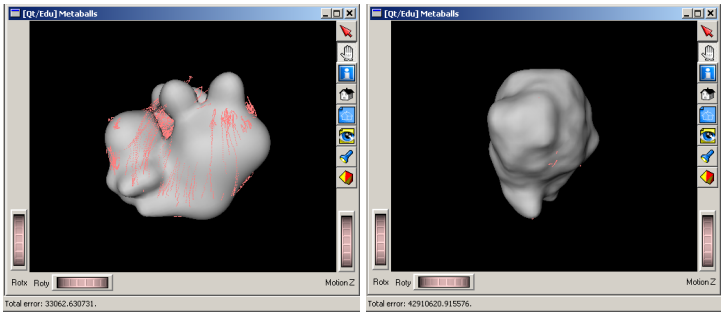
6.2.3.2 Gleno-humeral and elbow joint implicit surfaces

Applying the method laid out in Section 6.2.2 to the gleno-humeral joint quaternions and the elbow Euler angles, we obtain the voxelisations and iso-surfaces shown in Figure 6.49.

In the case of the elbow, as rotation is not possible in one of the planes, the corresponding Euler angles are two-dimensional, therefore yielding a 2D voxelisation, as well as a 2D implicit surface.



(a) (b) (c)



(d) (e)

Figure 6.47: Least-squares fitting of an iso-contour to surface points extracted using various methods: (a) solid angles; (b) shrink-wrapping; (c) alpha-shapes; (d) Marching Cubes; (e) primitives-in-voxelisation.

If we wished to absolutely use a quaternion representation for the elbow rotations, we could of course always approximate the boundary of the data of Figure 6.10(b) by an alpha-shape complex. However, as mentioned before, we would like to avoid having to use an explicit representation, and furthermore this would not readily allow us to use the joint limits representation we are about to detail in the following section, which takes into account joint coupling.

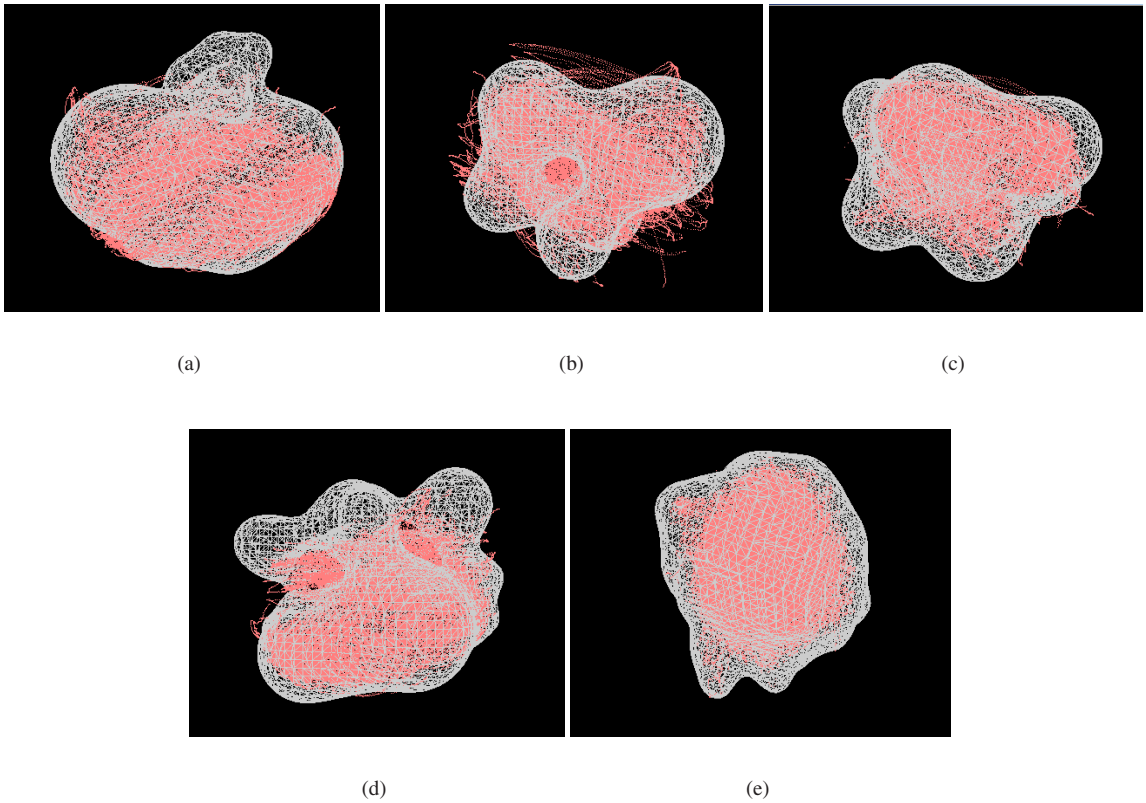
6.2.3.3 Sterno-clavicular and gleno-humeral joint limits

Here again, we apply the method of Section 6.36 to obtain Euler angle sterno-clavicular joint limits and quaternion joint limits for the gleno-humeral joint in the sterno-clavicular joint referential. The resulting voxelisations and iso-surfaces are shown in Figure 6.50, these representing the global joint limits for each joint, without coupling.

The joint inter-dependency between the sterno-clavicular and gleno-humeral joints originates in the shoulder complex structure, and is not a question of collision with body limbs, as was shown in Figure 3.11 to illustrate how upper arm elevation is dependent on clavicula elevation.

6.2.4 Hierarchical joint limits

The joint limits derived previously for the shoulder compound and the two coupled joint sets only capture the intra-joint dependencies. In order for inter-joint coupling to also be reflected, we need to define hierarchical joint



(a) (b) (c)

(d) (e)

Figure 6.48: Wire-frame iso-contours and surface points: (a) solid angles; (b) shrink-wrapping; (c) alpha-shapes; (d) Marching Cubes; (e) primitives-in-voxelisation.

limits for the coupled joints. We first describe our method on the case of the gleno-humeral and elbow joints, and then apply it in a similar way as for the sterno-clavicular and gleno-humeral joints.

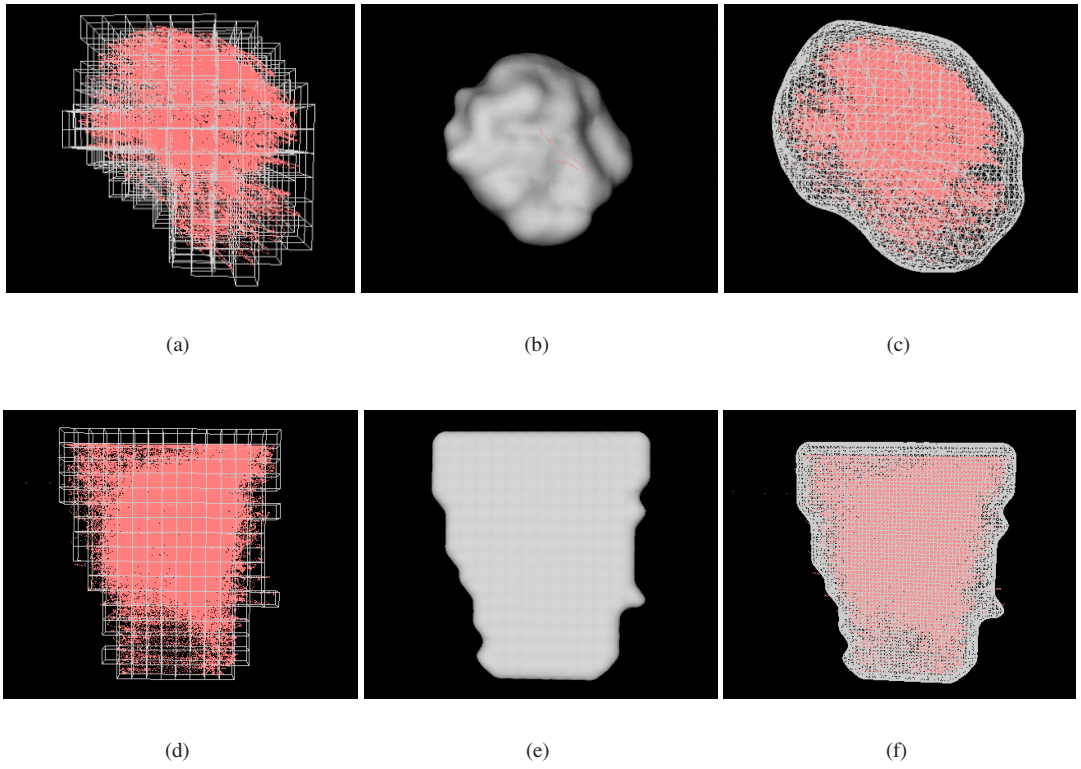
6.2.4.1 Gleno-humeral and elbow hierarchical joint limits

The idea outlined in the previous chapter was that for each sub-set of rotations of the parent joint, there is a defined set of acceptable rotations for the child joint. Of course, it is out of the question to define such a range of motion for the elbow joint for each individual rotation of the gleno-humeral joint. In motion capture terms, this would be totally infeasible. We recall that presently, to each rotation of the gleno-humeral joint, there is an associated elbow joint rotation. If we divide the space of gleno-humeral joint rotations into 50-or-so clusters, the number of data points from which to derive joint limits for the elbow joint is relatively low in each cluster.

In practice, we take those clusters to be positive voxels of a voxelisation of the gleno-humeral rotation quaternion. However, contrary to our method for retrieving an implicit surface, we do not want to sub-divide the space too finely, as we would otherwise run into the afore-mentioned problem of under-sampling. Therefore, we have performed a $7 \times 7 \times 7$ voxelisation on the gleno-humeral data, as shown in Figure 6.51.

For each positive voxel of the parent joint voxelisation, which we will call a keyframe voxel, we need to compute the implicit surface of joint limits for the child joint.

As for each quaternion of the gleno-humeral joint we have the associated Euler angle of the elbow joint, the data points of the elbow for each keyframe voxel are readily obtained. As was to be expected, in spite of the fact that long motion capture sequences have been used, featuring hundreds of thousands of samples, we still



(a) (b) (c)

(d) (e) (f)

Figure 6.49: Joint limits for the gleno-humeral and elbow joints: (a) voxelisation of the gleno-humeral joint quaternions; (b) extracted implicit surface; (c) wire-frame implicit surface and data; (d) voxelisation of the elbow joint Euler angles; (e) extracted flat implicit surface; (f) wire-frame implicit surface and data.

experience the problem of under-sampling. This is clearly visible in Figure 6.52, where we show the Euler angle data corresponding to four of the most populated keyframe voxels. It is obvious that it will be close to impossible to derive a boundary implicit surface from such sparse data.

Assuming that the variation in child joint limits is slight from one parent keyframe voxel to another, we group our data using all the nearest neighbours. In other words, to the Euler angle data corresponding to a given keyframe voxel, we also add the Euler angle data from the direct neighbour keyframe voxels in the 26-connectivity sense.

In this manner, we obtain sub-clouds of data that are much more populated, and that are nevertheless nearly identical in general shape to the original data. These new data clusters can now be approximated by an implicit surface as per the method described in Section 6.2.1, meaning that we voxelise the data as finely as we wish, fill all the positive voxels with spherical primitives and then compute the corresponding iso-contour.

To summarise, we have performed the following operations for our two joints:

1. the gleno-humeral joint quaternion data is finely voxelised, the positive voxels are filled with spherical primitives and an iso-surface is derived;

2. the gleno-humeral joint quaternion data is more coarsely voxelised in order to establish the keyframe voxels;

3. an implicit surface is fitted to the elbow data corresponding to each keyframe voxel using the same algorithm as in step 1.

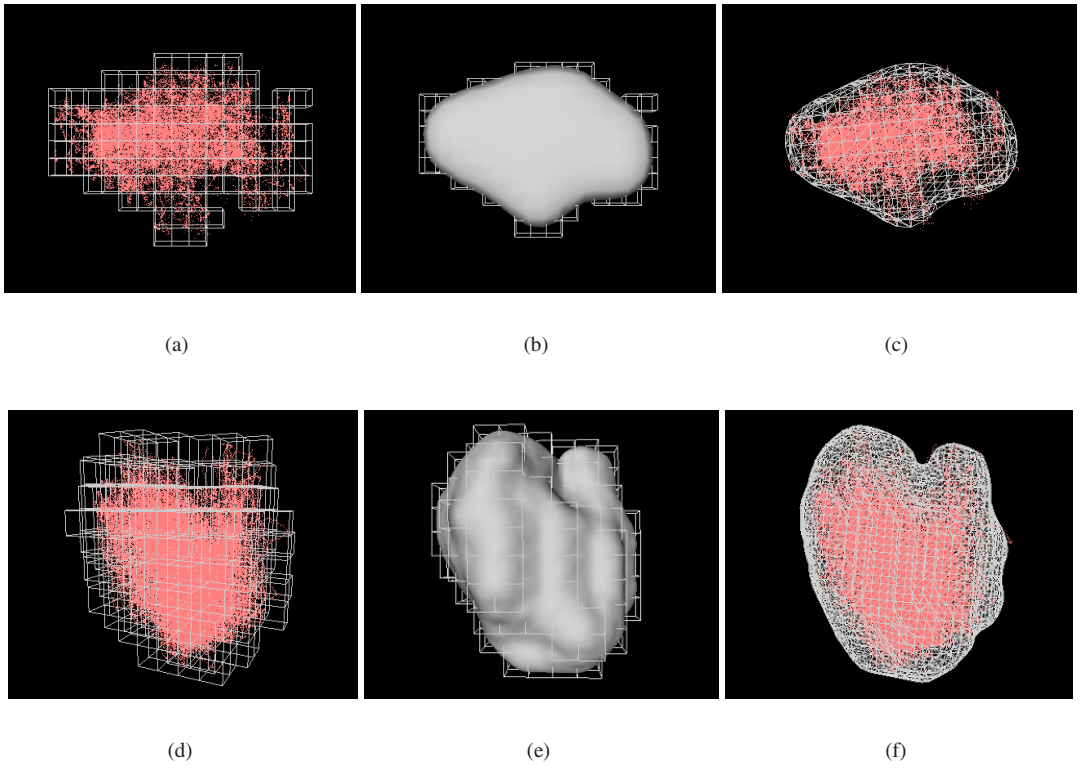


Figure 6.50: Joint limits for the sterno-clavicular and gleno-humeral joints: (a) voxelisation of the sterno-clavicular joint Euler angles; (b) extracted flat implicit surface; (c) wire-frame implicit surface and data; (d) voxelisation of the gleno-humeral joint quaternions; (e) extracted implicit surface; (f) wire-frame implicit surface and data.

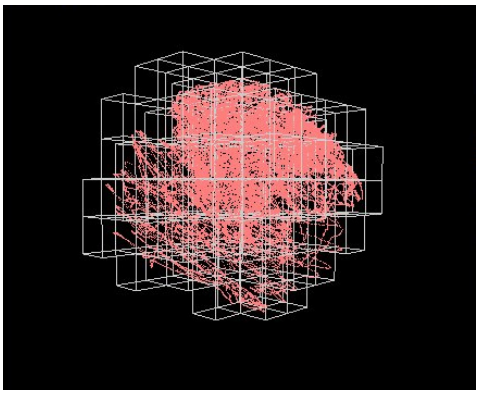


Figure 6.51: Low-resolution voxelisation of the gleno-humeral joint data.

As the initial voxelisation of the parent joint is coarse, the joint limits of the child joint, from one keyframe voxel to its neighbour can be more different than would be desirable. Even if the changes in shape are slight, when crossing from one keyframe voxel to another, the change in shape of the child joint limits might not be smooth enough for some applications, such as animating virtual humans. Say the humanoid is in a posture where the elbow joint is at its limit, or even outside the valid joint limits and was therefore projected onto the implicit

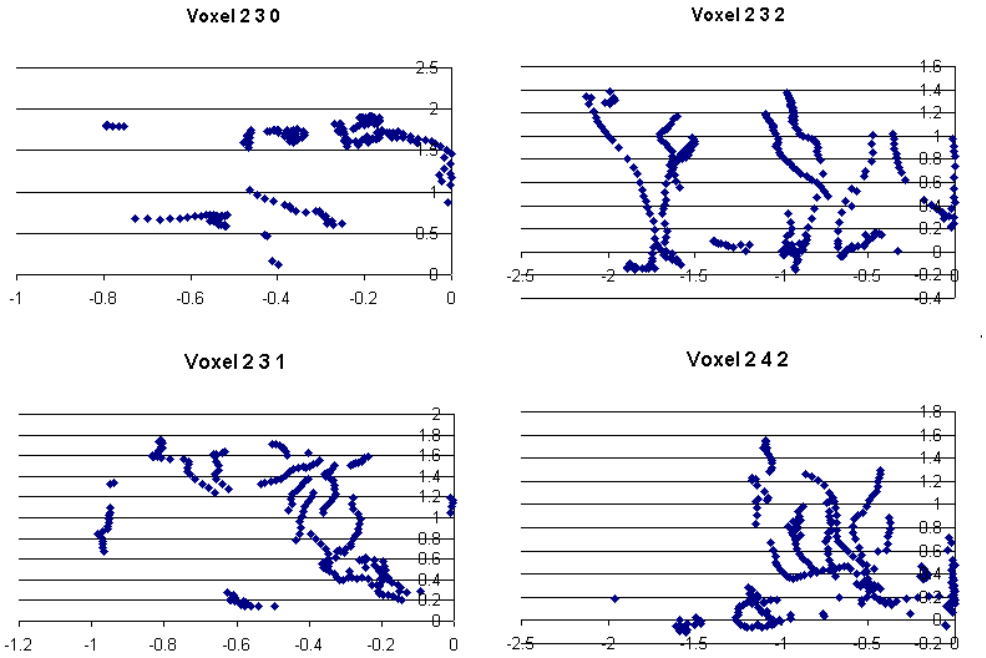


Figure 6.52: Euler angle data of the elbow joint for the most populated keyframe voxels of the gleno-humeral joint data.

surface to determine its closest valid rotation. If we slightly change the rotation at the gleno-humeral joint, we might also move from one keyframe voxel to another. If the corresponding elbow joint limits are too different, the new clamped elbow position might vary sufficiently for the animation, as it may appear jerky.

To prevent this, we insert intermediate keyframe voxels for which we compute the child joint limits by morphing the original keyframe voxels. Such insertion of voxels is performed between horizontally, vertically and diagonally neighbouring keyframe voxels, in other terms we "explode" the initial keyframe voxelisation to insert new keyframe voxels, as shown in Figure 6.53. To compute the intermediate joint limits, we will morph the two implicit surfaces corresponding to either side of the newly created keyframe voxel.

A plethora of methods exists for volume-based morphing using interpolation of the skeletal primitives. Many of them are presented in [Lazarus and Verroust1998], and in recent years, several have been proposed, such as the level-set approach [Breen and Whitaker2001], the radial-basis function method [Morse *et al.*2001], region correspondence [Treece *et al.*2001] or variational implicit functions [Turk and O'Brien1999]. However, all of these methods are relatively complicated and address the problem of morphing between fundamentally different shapes. We, on the other hand, are dealing with implicit surfaces that only differ slightly, and a linear morphing algorithm should be sufficient to handle this case.

We have chosen to implement an interpolation scheme similar to [Ranjan and Fournier1995] that morphs between unions of spheres, as shown in Figure 6.54 by constructing a bipartite graph between the primitives of source object A and target object B.

The weights associated to the edges of the graph represent the distance between a primitive *a* of shape A and a primitive *b* of shape B, and this distance is defined as:

$$d(a,b) = [(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2] + (e_a - e_b) \tag{6.9}$$

where (x_a, y_a, z_a) is the centre and e_a the thickness of primitive a and (x_b, y_b, z_b) is the centre and e_b the thickness of primitive b.

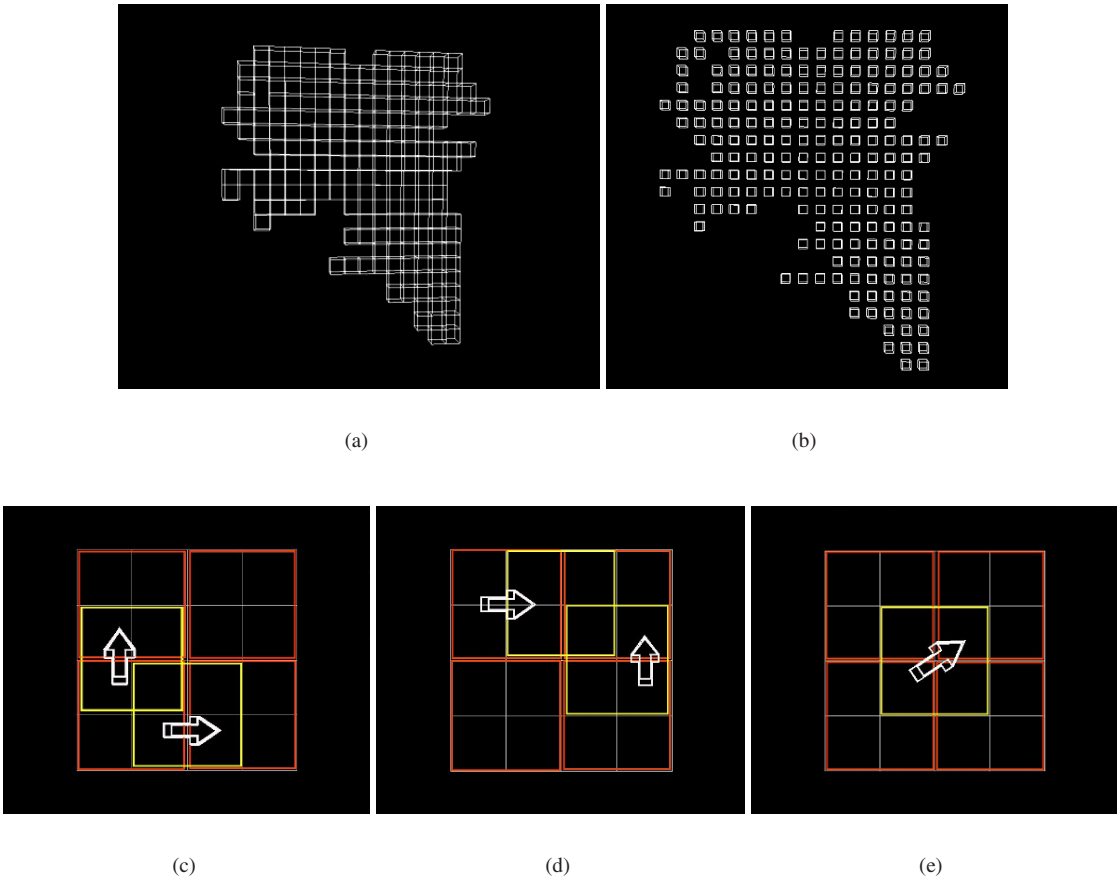


Figure 6.53: Voxelisation exploding and voxel insertion illustrated in 2D: (a) example of a voxelisation; (b) exploded voxelisation, ready for insertion of intermediate voxels; (c) horizontal insertion; (d) vertical insertion; (e) diagonal insertion.

As shown in Figure 6.55(b), each primitive from the source object is matched to exactly one primitive of the target object, so that the total cost of the matching is minimal, the cost being defined as $\sum d(a.b)$ where $a \in A, b \in B$.

We have modified the original algorithm slightly so as not to have to compute the maximum bipartite graph matching, that is to find the matching of minimal cost. For each primitive of source shape A, we simply find the primitive in target shape B that is the closest with respect to the distance notion of eq.(6.9), but still holding true the condition that each primitive in A is uniquely matched to a primitive of shape B. As the numbers of primitives in shapes A and B are not necessarily equal, we perform a second pass for the remaining primitives, without condition this time, as the remaining primitives of the first shape can be matched to any primitive of the second shape, depending on which has the left-over primitives.

Once this primitive matching has been established, we just need to interpolate between the centres and radii of the matched primitives, over the chosen number of interpolation steps .

By proceeding in this fashion, we get a smooth interpolation for our elbow joint limits, and an example of an interpolation in 10 steps is shown in Figure 6.56.

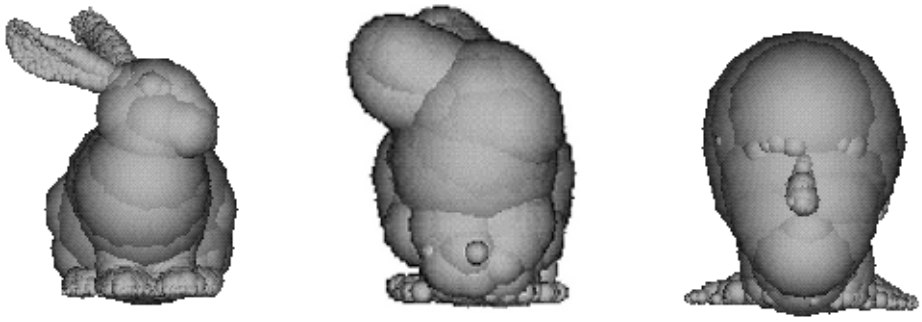


Figure 6.54: Shape interpolation with the shapes represented as unions of spheres [Ranjan and Fournier1995].

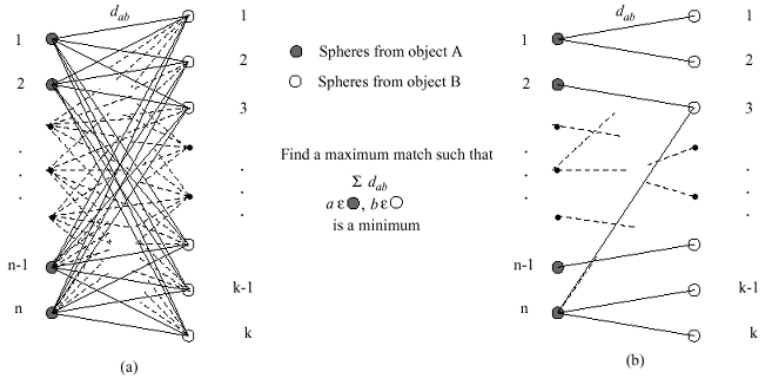


Figure 6.55: Shape interpolation with unions of spheres [Ranjan and Fournier1995]: primitive matching using a bipartite graph.

6.2.4.2 Sterno-clavicular and gleno-humeral hierarchical joint limits

To obtain hierarchical joint limits, we proceed in an identical fashion as for the gleno-humeral and elbow joints, the only difference being that the parent keyframe voxels are 2-dimensional and the child joint limit implicit surfaces 3-dimensional. We applied a 10×10 voxelisation on the sterno-clavicular data, as shown in Figure 6.57.

We apply the various steps of the algorithm described in Section 6.2.4 to obtain an exploded voxelisation for the sterno-clavicular joint, as well as the intermediate keyframe voxels. For each such voxel, we compute the corresponding gleno-humeral joint limits by morphing, for which we show the results in Figure 6.58.

The scheme we have just finished describing can of course be extended to any joint of the body, insofar as the rotations of the joint can be measured. If this condition is fulfilled, then a measurement protocol needs to be set up, and once the rotations derived, joint limits for that joint or for a set of joints can easily be computed using our technique, all within a few minutes in one integrated software.

6.3 Discussion

In this section, we will briefly summarise our suggestions for improvement of the quality of the joint limits resulting from the entire process, these improvements residing at the data collection level.

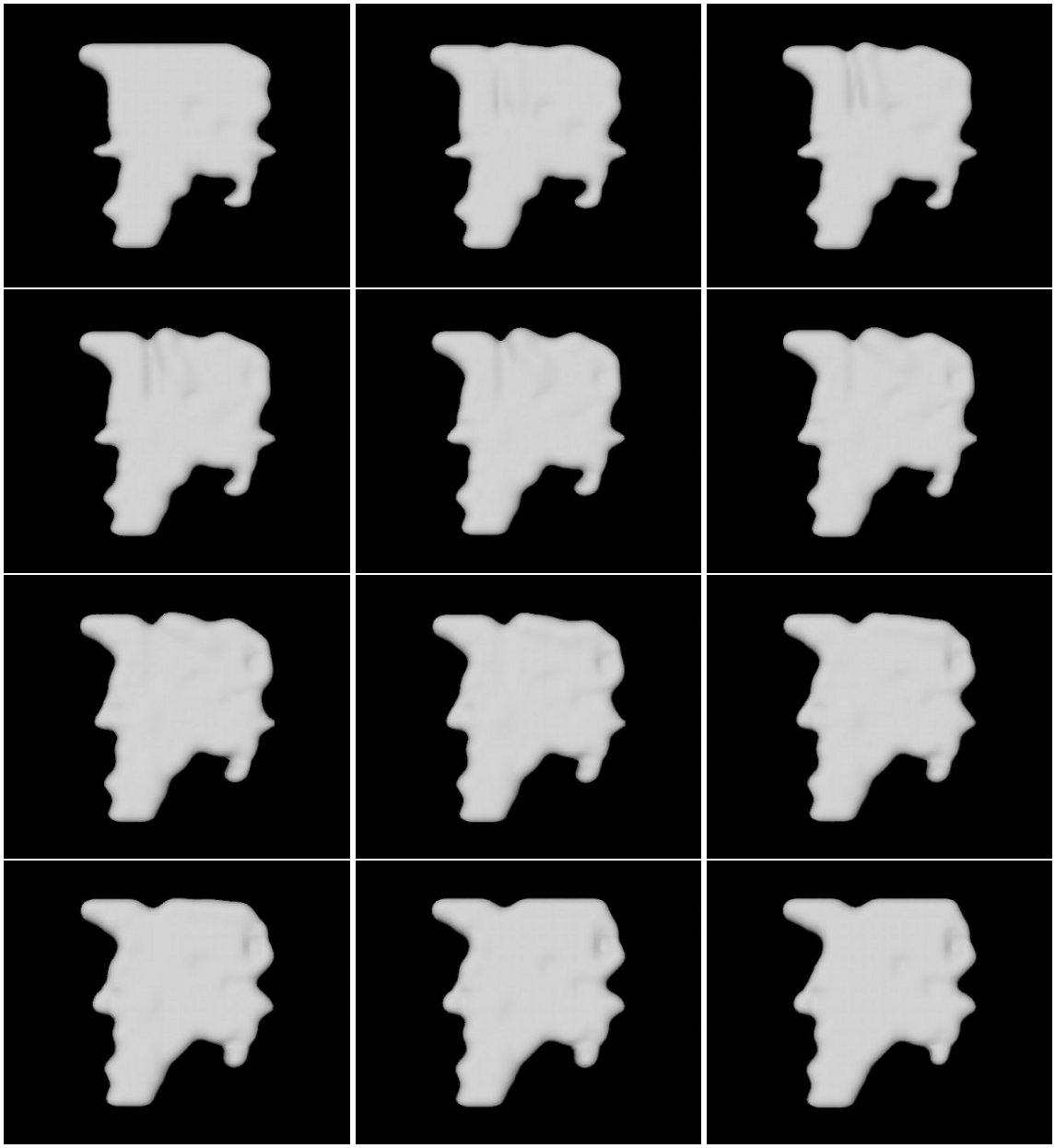


Figure 6.56: Interpolation of two elbow joint limits, in 10 intermediate steps.

6.3.1 Data completeness

The main drawback of our method is that the quality of the data we use to create our representation is key to its accuracy. The current acquisition process relies on optical motion capture and when sampling the range of motion of a joint, we have no immediate feed-back on whether we have effectively sampled the entire attainable space.

Under-sampling is not really a problem, as even sparsely sampled regions will be included in the voxelisation, and thus, in the implicit surface boundary. However, if some areas are never sampled, in spite of the fact that they

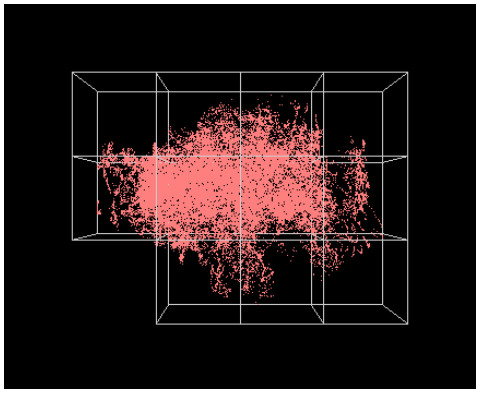


Figure 6.57: Voxelisation of the sterno-clavicular joint data.

can be reached, this will be reflected in the joint limits, and might cause havoc during human body tracking or character animation by rejecting perfectly valid postures. To remedy this problem, one could consider designing an application that provides immediate visual feed-back directly during motion acquisition. In this fashion, the current trial-and-error manner of performing data collection could be avoided and greatly increase the speed and efficiency of this part of the process. We were not able to do so in our case, as the commercial motion capture package we were working with did on the one hand not allow the creation of plug-ins or other software add-ons, and on the other, required a certain amount of post-processing, thus shattering any dream of real-time feed-back during data acquisition.

6.3.2 Data accuracy

Optical motion capture requiring reflective markers to be placed on the skin, or the actor to wear a tight suit, skin-to-bone displacement will always be present thus adding some uncertainty to the precision of the measures. How acute this phenomenon is depends on the measured joint, but its presence is inevitable to some extent. A possible solution would be to use a more reliable data collection method, or to have access to usable clinical data. Another possibility would be to establish a precise, rather than approximate, motion model of the skin-to-marker displacements so as to take these into account when collecting joint rotations.

6.4 Data validation

In order to assert the validity of our ranges of motion, we will now compare them to the values in the literature, whenever such a comparison is possible. It is of course not our objective to prove that our joint limits obtained from motion capture are as precise as those published in the specialised biomechanical press. Clinical observations are precise to the millimetre, and aim at providing valuable information to the medical corps, for detecting pathologies, for example.

Our joint limits do not have the ambition of being rigorously exact, and they suffer from a certain amount of accumulated errors present at the following levels:

- marker reconstruction by the Vicon;
- computation of the rotations from the reconstructed markers;
- approximation of the data points by an implicit surface.

In any case, for validation purposes, it is necessary for the joint limits representations to be comparable, i.e. the reference joint limits should be formalised in a way that readily allows us to test our limits against them. We

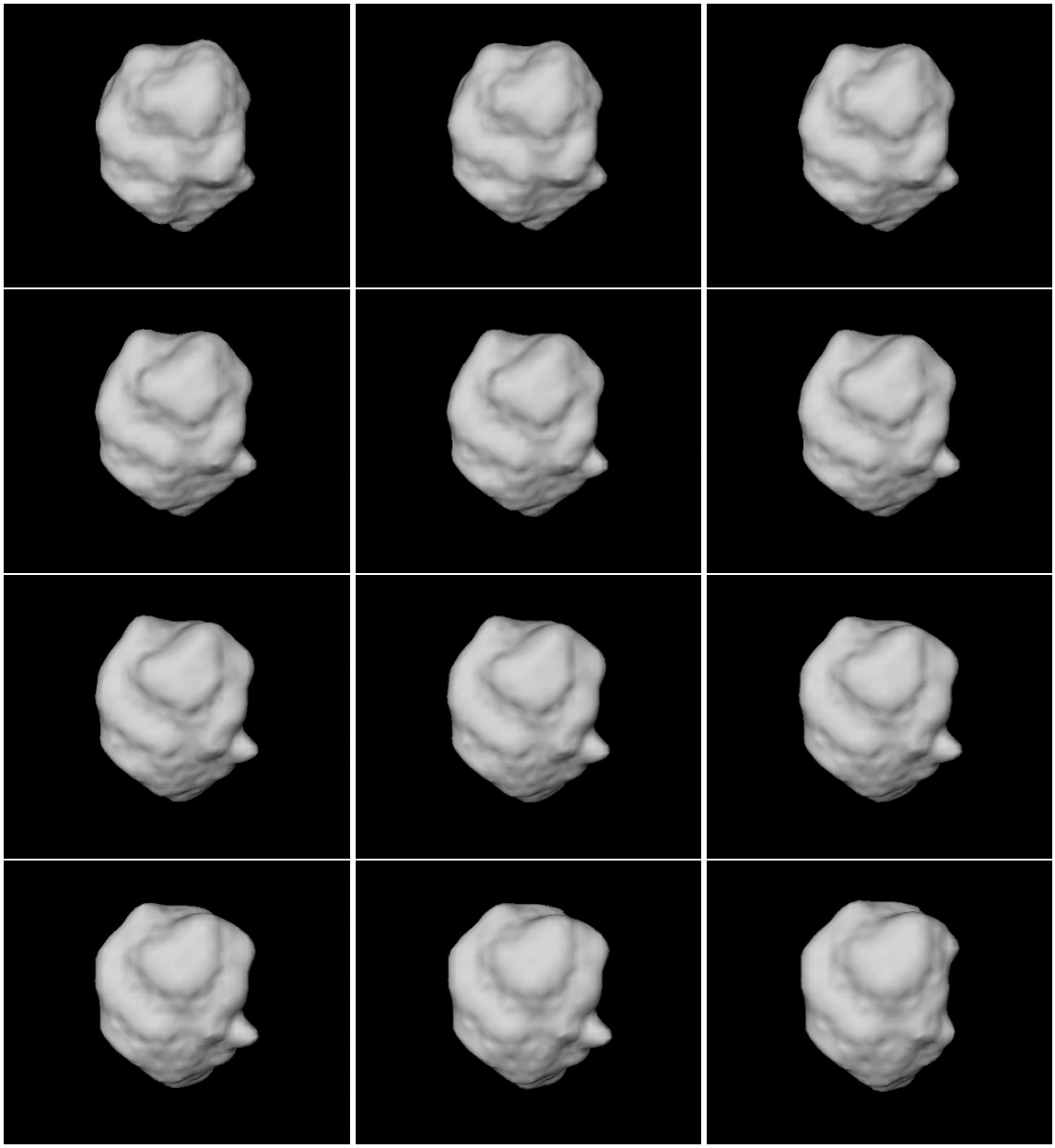


Figure 6.58: Interpolation of two elbow joint limits in 10 steps. (a) is the source shape and (l) the target shape.

listed the existing joint limit representations in Section 5.2, these being minimal and maximal values on Euler angles, spherical polygons, joint sinus cones and triangular Bézier patches. With the exception of the latter, data is available in the literature, thus allowing comparisons with our model.

6.4.1 Joint sinus cones and spherical polygons

As we pointed out in Section 5.2, the most precise existing joint limit formulations in computer graphics are joint sinus cones [Engin and Tümer1989a] and spherical polygons [Korein1985]. Joint sinus cones limit the range of

motion of a joint by a truncated cone whose tip is located at the joint and whose base defines the boundaries of angular motion. Spherical polygons define the boundary of motion on a unit sphere centred in the joint, this boundary being represented by a spherical polygon on the surface of the sphere. Both representations have the disadvantage of only constraining swing, thus discarding the twist component.

6.4.1.1 The shoulder compound

Based on the shoulder compound joint sinus cone base detailed in [Engin and Tümer1989b], the spherical polygon was designed at the Virtual Reality Laboratory of the EPFL, and implemented in the proprietary animation library *libhbody*.

This spherical polygon being defined as a chain of points on the surface of a unit sphere, we can expand these positions so that they are outside the boundaries of the implicit surface, then consider these positions as unit quaternion keyframes and interpolate them to obtain a keyframe sequence. When playing this keyframe sequence, all orientations will be clamped to the closest point on the implicit surface, which gives us the boundaries in terms of arm elevation. By comparing these to the sequence obtained by moving along the spherical polygon border, we can estimate whether our joint limits are correct in terms of abduction/adduction and flexion/extension, as well as in terms of continuity.

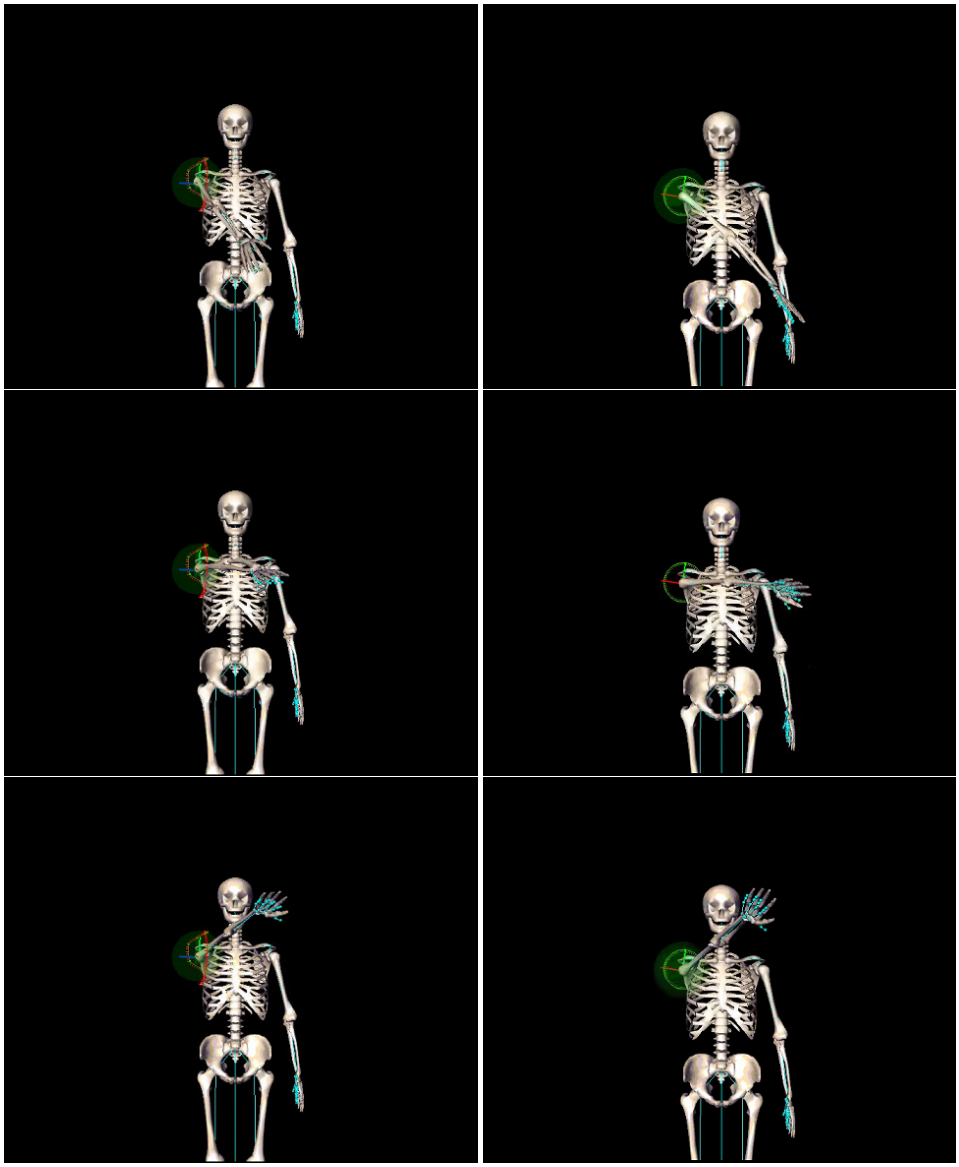


Figure 6.59: The left column shows the motion of the arm when moving along the border of the spherical polyon, and the right column the equivalent motion when being constrained by our implicit surface.

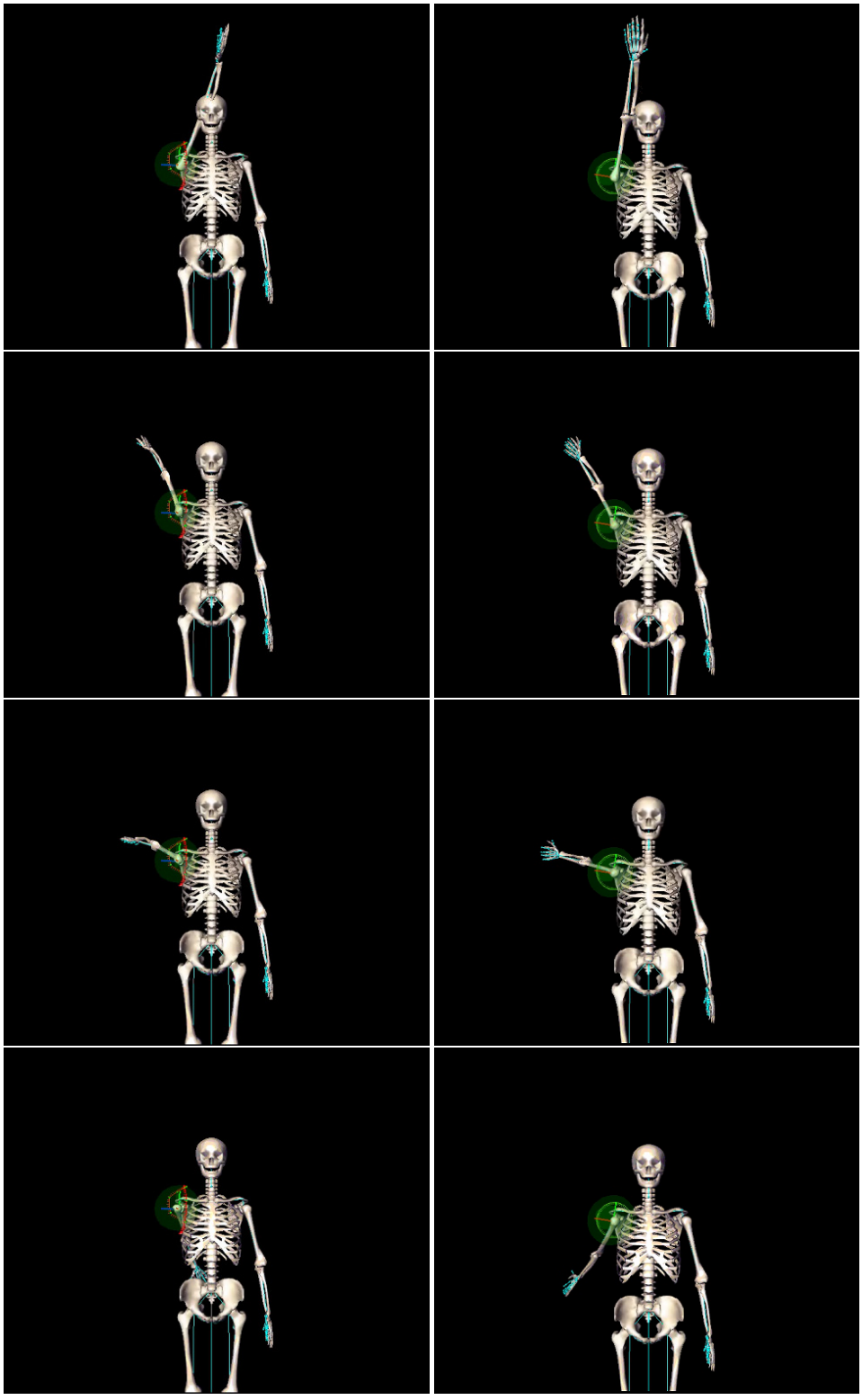


Figure 6.60: Figure 6.59 continued.

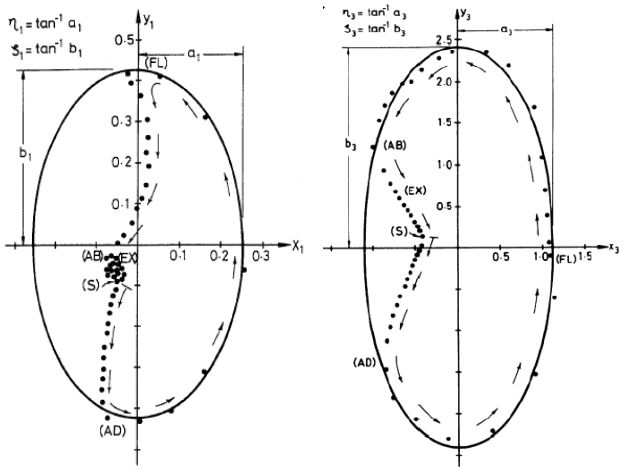


Figure 6.61: Cross-sections of the (a) sterno-clavicular and(b) gleno-humeral joint sinus cones.

Judging by the results obtained and shown in Figures 6.59 and 6.60, we can conclude that the respective boundaries of the two joint limits representations are close, except in some areas where the influence of the measurement process can be noted. This is the case for the limit at maximal abduction, where in the case of the spherical polygon, the arm penetrates the head. Given that the motion capture actor could not remove his head for the duration of capture, the arm moves around the head instead of through it. In the same manner, in the last frame of the sequence, when the arm is fully extended, in the spherical polygon case, the arm is bent far backwards and penetrates the thorax. This is due to the fact that at the skeleton level, the shoulder joint indeed has such mobility. However, in the case of a human being not only of bone but also of flesh, the amount of extension is limited by its physical boundaries. These boundaries can be extended if pressure is exerted on the limb to push it to its limit, but such a posture is not only uncomfortable, but also difficult to reach during motion capture session.

The alert reader may remember that spherical polygons are only boundaries for angular rotation, i.e. the twist component is not present. This is the case for all joint limits representations that could serve as reference, so no global assessment of our derived joint limits is possible.

In comparative terms, one can say that our representation reflects in a more realistic way the actual range of motion of a live human being, taking into account the various factors that limit a range of motion, These would typically be the physical presence of the body and the fact that muscle-driven motion has a shorter range than passive motion where pressure is exerted.

The sterno-clavicular and gleno-humeral joints The ellipsoidal-like bases of the sterno-clavicular and gleno-humeral joint sinus cones are depicted in [Engin and Tümer1989c], and we reproduce these drawings in Figure 6.61. These were obtained originally by making a subject perform a circumductory arm motion along the natural boundaries of upper arm motion, thus passing through the points of maximal abduction, extension, adduction and flexion. The unit vectors from sterno-clavicular joint to acromio-clavicular joint, and from gleno-humeral to elbow joint were thus sampled. The plane perpendicular to the average of these vectors is then computed, at a unit distance from the origin. The intersection of all the vectors with this plane yields the discreet points shown in Figure 6.61 from which the cone base was then derived.

The two axes of each cone base are directly related to our motion components of the joints, and the apex

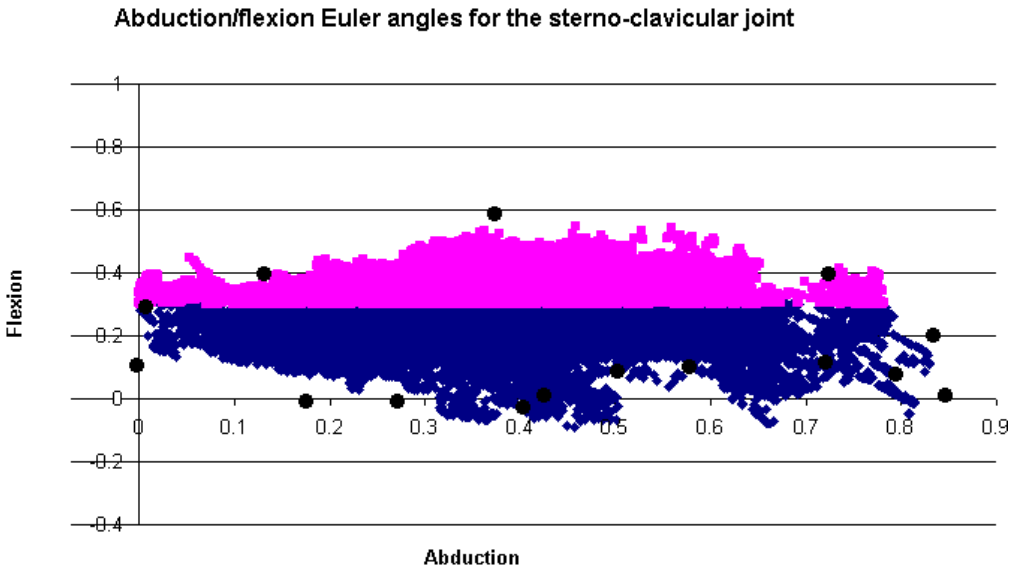


Figure 6.62: Projection of the cone base data points onto the derived Euler angle values of the sterno-clavicular joint.

values are also proportional to the range of motion available in each direction. There is of course a scaling factor as the minimum and maximum values on each axis of Figure 6.61 do not represent actual Euler angle values, but only cone base height and width.

In the case of the sterno-clavicular joint, we can directly use the abduction/adduction and flexion/extension values that we computed and showed in Figure 6.17(b). We project the discreet points forming the cone base of Figure 6.61(a) onto the plot of Euler angles, after applying the appropriate scaling. The result is shown in Figure 6.62. The two shapes compare relatively well, keeping in mind that:

- our data was not collected uniquely on the boundaries of motion, but over the entire range of motion, and we therefore do not have a smooth set of points representing the limits in terms of angular motion;
- our measurements were subject to skin-to-bone displacement, and that in spite of the correction described in Section 6.1.2, we cannot expect to eliminate all errors.

We proceed in a similar fashion for the gleno-humeral joint, after converting our quaternion rotations to Euler angles. As we have mentioned before, this decomposition is not unique. We have therefore computed the equivalent swing and twist components, as per [Baerlocher and Boulic2000], and kept only the swing component, given that the cone base represent limits in terms of limb orientation only. Decomposing a quaternion into swing and twist boils down to finding the swing axis-angle $[s_x \; s_y \; 0]^T$ followed by an axial rotation of magnitude ϕ around the z-axis, i.e. axis-angle $[0 \; 0 \; \phi]^T$. From the original quaternion (q_x, q_y, q_z, q_w), if $q_z \neq 0$ and $q_w \neq 0$, we have:

$$\phi = 2. \arctan\left(\frac{q_z}{q_w}\right)$$

The swing values can then be computed as follows:

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \frac{2}{\left(\frac{\sin\beta}{\beta}\right)} \cdot \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix} \cdot \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$

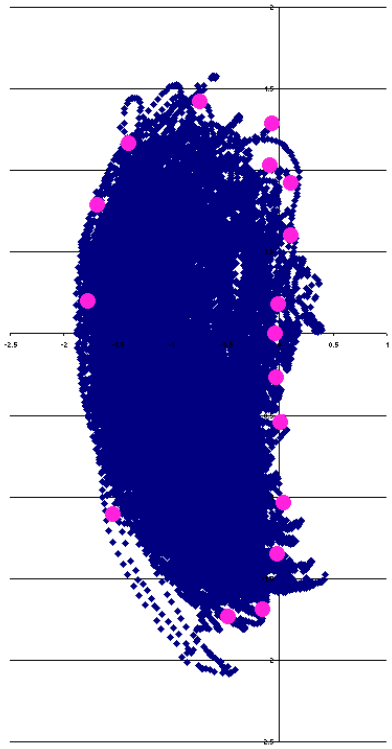


Figure 6.63: Projection of the cone base data points onto the derived swing values of the gleno-humeral joint.

where $\gamma = \frac{\phi}{2}$and $\beta = \arctan\left(\frac{\sqrt{q_x^2+q_y^2}}{\sqrt{q_z^2+q_w^2}}\right)$ under the constraint that $0 \leq \beta \leq \frac{\pi}{2}$ and $-2\pi \leq \phi \leq 2\pi$.

We also convert the discreet cone base points, after scaling, to their equivalent swing values. The superposed data plots of the two point sets are shown in Figure 6.63. Here again, the shape of the cone base roughly corresponds to the outline of our angular motion data points.

6.4.2 Independent axis rotation

If we display the 3D quaternions representing the boundary of an implicit surface centred in the origin of the referential representing the joint rotation centre, then the two intersection points of the surface with the x-axis can be considered as the range of motion for abduction/adduction, the two intersection points on the y-axis as flexion/extension and finally the two intersection points on the-z axis as the range of axial rotation. This is equivalent to applying our joint limits to a model in an animation package, and to observing at what rotational values along each axis the limits are reached.

6.4.2.1 The shoulder compound

The values obtained either way can then be compared to published ones [Luttgens and Hamilton1997] for the shoulder compound, i.e. the range of motion of the upper arm. The values in the afore-mentioned publication are based on four different sources, and the shoulder rotation components are summarised in Table 6.2.

We would like to remind the reader that in the case of the shoulder compound and the gleno-humeral joint, the notions of flexion and extension follow anatomical usage, which we explained in Section 3.2.1. In the computer graphics and animation convention, flexion/extension means rotation in the transversal plane, whereas in the

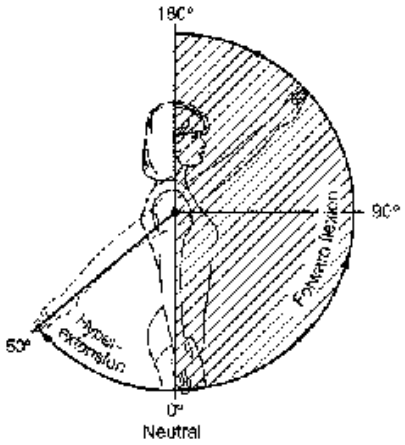


Figure 6.64: Anatomical convention on the definition of flexion and extension.

Movement	Range of motion in degrees	
	Luttgens & Hamilton	Present work
Abduction	170-180	170
Flexion*	130-180	180
Extension*	30-80	35
Internal rotation	60-90	72
External rotation	70-90	90

Table 6.2: Average ranges of motion.

anatomical convention, it means rotation in the sagittal plane, as illustrated in Figure 6.64. In the tables of this section, if we write flexion* or extension*, this indicates that we are referring to the anatomical definition of these terms. Otherwise, it is understood that we mean flexion or extension in the computer graphics terminology.

The range of motion varies enormously from one source to another in the case of flexion and extension. It should be pointed out that Luttgens and Hamilton's measurement protocol is based on external observations of limb motion, and not on any sophisticated techniques such as those used by [Engin and Tümer1989a], [Engin and Chen1986] or [VanDerHelm and Pronk1995].

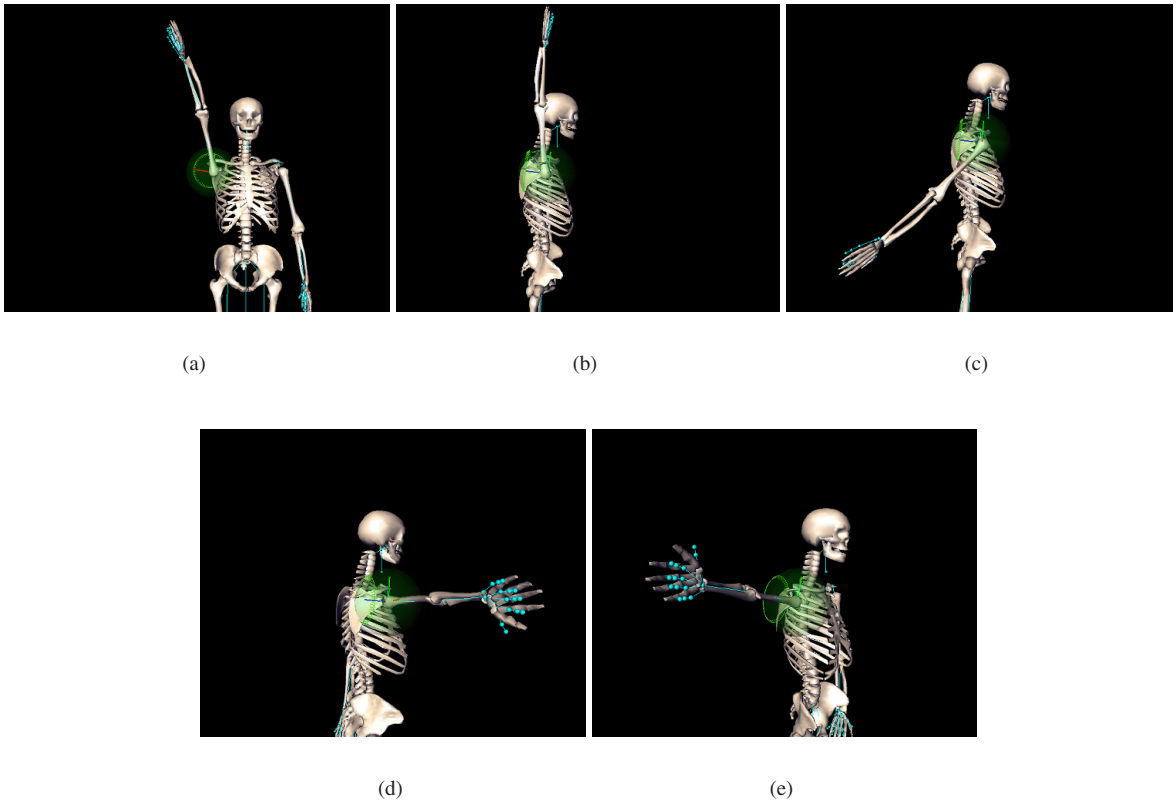
For a visual illustration of the range of motion along each axis provided by our implicit surface joint limits, see Figure 6.65. One can note the absence of the adduction component. In terms of pure joint motion, adduction at the shoulder joint (and gleno-humeral joint) is possible, but for a living human being, performing pure adduction is impossible, as the arm will naturally be limited by the presence of the body. We will therefore forget about adduction altogether, as we consider it non-measurable.

6.4.2.2 The gleno-humeral, sterno-clavicular and elbow joints

The reference data for the gleno-humeral, sterno-clavicular and elbow joints will be taken from the Appendix of [Maurel1998], these reference values as well as the corresponding local referentials being shown in Figure 6.66.

Reference values for the elbow were also published by Luttgens and Hamilton, and are shown in Table 6.3, where forearm pronation and supination are what we have modelled as elbow twist.

For 3D joints, we compute the intersections with the implicit surface and the referential axes, but for 2D joints, such as the sterno-clavicular and elbow, we can simply retrieve the values of the range of motion from the



(a) (b) (c)

(d) (e)

Figure 6.65: Range of motion on each axis and in each direction, for the shoulder compound, as defined by the implicit surface joint limits: (a) abduction; (b) flexion*; (c) extension*; (d) inner axial rotation (thumb down); (e) outer axial rotation (thumb up).

Movement	Range of motion in degrees
Elbow flexion	140-145
Elbow extension	0-10
Forearm pronation	80-90
Forearm supination	80-90

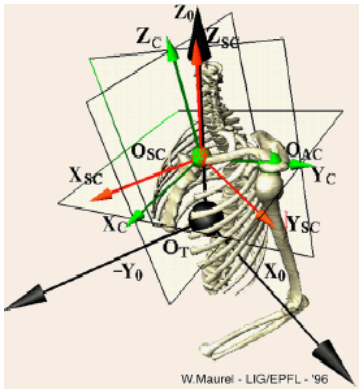
Table 6.3: Average ranges of motion.

Euler angle plots in Figure 6.12(b) and Figure 6.17(b), and compare them to the reference values of the literature.

It should be noted that in most studies, elbow extension is considered to be zero, and this is an assumption we have also made in the course of our measurements.

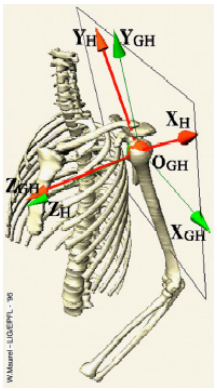
The comparative values for each joint are displayed in Tables 6.4, 6.5 and 6.6, given that the initial pose is also as per the anatomical definition, i.e. hand alongside the body.

In the case of the elbow, and more specifically its twist component, it must be pointed out that, as shown in the table of Figure 6.66(d), the range of motion is of $180\,\mathrm{deg}$, knowing that at the initial position, ulno-radial rotation is reported by [Maurel1998] to be of about $100\,\mathrm{deg}$. Given that our model does not explicitly represent the ulnar and radial bones, but expresses this rotation as a twist component at the elbow level, and given that we do not know the initial position of these bones, the range of motion observed in the plot of Figure 6.12(b) is subsequently re-centred to yield an equal range of motion for inner and outer twist. It should be pointed out that



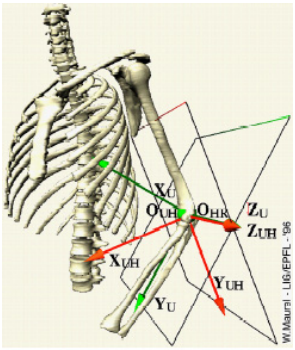
angle	θ_C	φ_C	ψ_C
axis	Z_C	X_C	Y_C
min	- 5°	+0°	-30°
max	+30°	+40°	+10°
rest	+22°	+24°	+ 0°

(a)



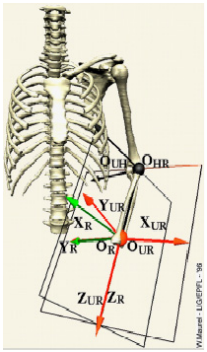
angle	θ_H	φ_H	ψ_H
axis	Z_H	X_H	Y_H
min	-40°	-100°	-90°
max	+100°	+50°	+90°
rest	-13°	+3°	+0°

(b)



angle	θ_U	min	max	rest
axis	Z_U	-140°	+0°	-57°

(c)



angle	θ_R	min	max	rest
axis	Z_R	+0°	+180°	+100°

(d)

Figure 6.66: Local referentials and limits: (a) sterno-clavicular joint; (b) gleno-humeral joint; (c) ulno-humeral joint, i.e. flexion of the elbow; (d) ulno-radial joint, i.e. twist of the elbow.

Movement	**Range of motion in degrees**	
	Maurel	Present work
Abduction	40	44
Adduction	0	0
Flexion	30	32
Extension	5	6

Table 6.4: Sterno-clavicular joint average ranges of motion.

according to [Biryukova *et al.*2000], joint geometry is relatively invariant from one subject to another in the case of the shoulder complex, but that this is not the case for the elbow joint. Their recordings show that the initial ulno-radial rotation varies from 71 to 108 deg for the seven subjects measured.

The complete range of axial motion adds up to 170 deg, 10 deg short of the reference range. The error is once again due to the lack of correlation between bone motion on the one hand, and soft tissue and skin motion on the other. Indeed, as shown in Figure 6.9, the amount of twist was measured by computing the angle

Movement	Range of motion in degrees	
	Maurel	Present work
Abduction	100	105
Flexion*	100	105
Extension*	50	43
Internal rotation	90	70
External rotation	90	90

Table 6.5: Gleno-humeral joint average ranges of motion.

Movement	Range of motion in degrees		
	Luttgens & Hamilton	Maurel	Present work
Flexion	140-145	140	143
Extension	0-10	0	0
Twist	180	180	170

Table 6.6: Elbow joint average ranges of motion.

between the elbow and wrist segments that are attached to, the upper and lower extremity of the forearm (see Figure 6.67). Ideally, to derive twist from these two vectors, the elbow segment would need to remain perfectly stationary. However, this is not what occurs, as the skin is pulled in the direct of the twisting motion, therefore also displacing the elbow segment, resulting in a damped output value with respect to the effective amount of twist performed. Such observations were already reported by [Cappozzo *et al.*1996], who state that skin marker displacements with respect to the underlying bone can range from a few millimetres up to 4 cm, the largest of such artefacts being most likely encountered for markers situated above anatomical landmarks in the vicinity of joints.

A visual assessment of the ranges of motion for each joint, on each axis, and in each direction is given in Figures 6.68, 6.69 and 6.70.

In short, the derived values are close to the reference literature, with a few minor discrepancies. This is not surprising since, in addition to the eventual measurement and approximation errors, some limits are uncomfort-

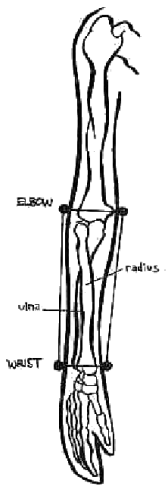
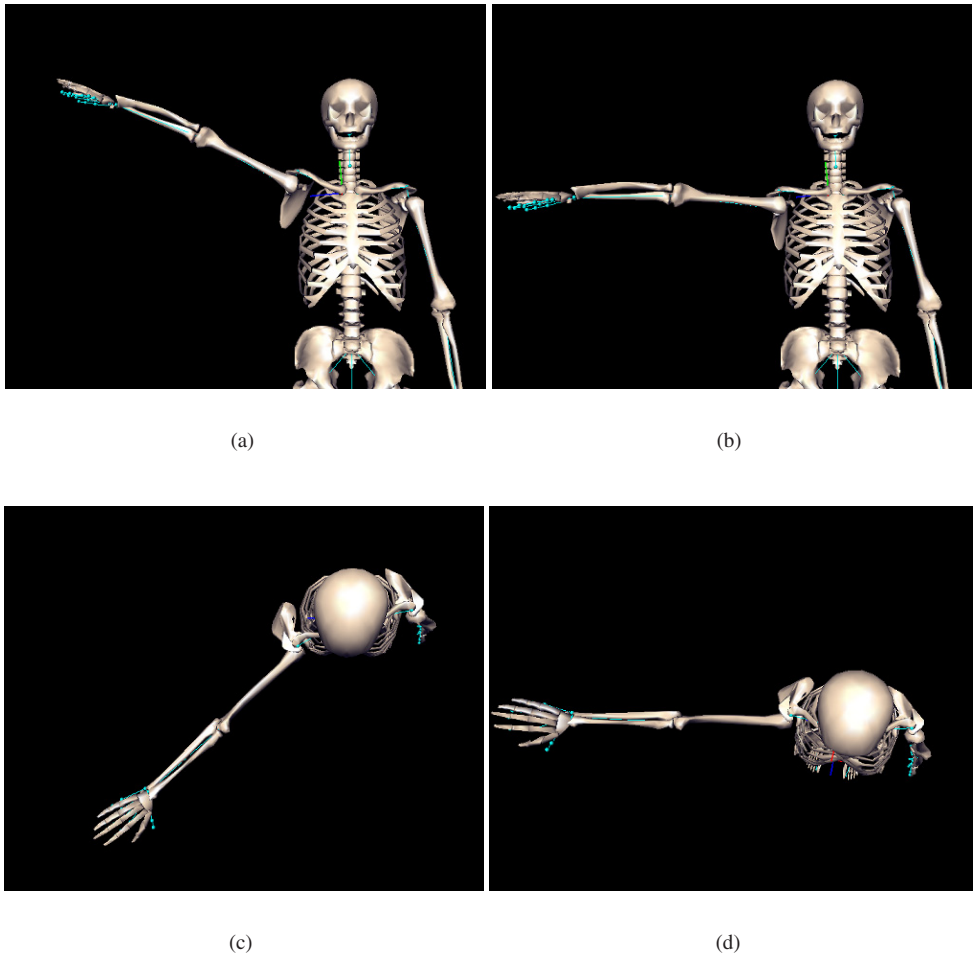


Figure 6.67: Configuration of the ulnar and radial bones, and of the motion capture markers [Menache1999].



(a) (b)

(c) (d)

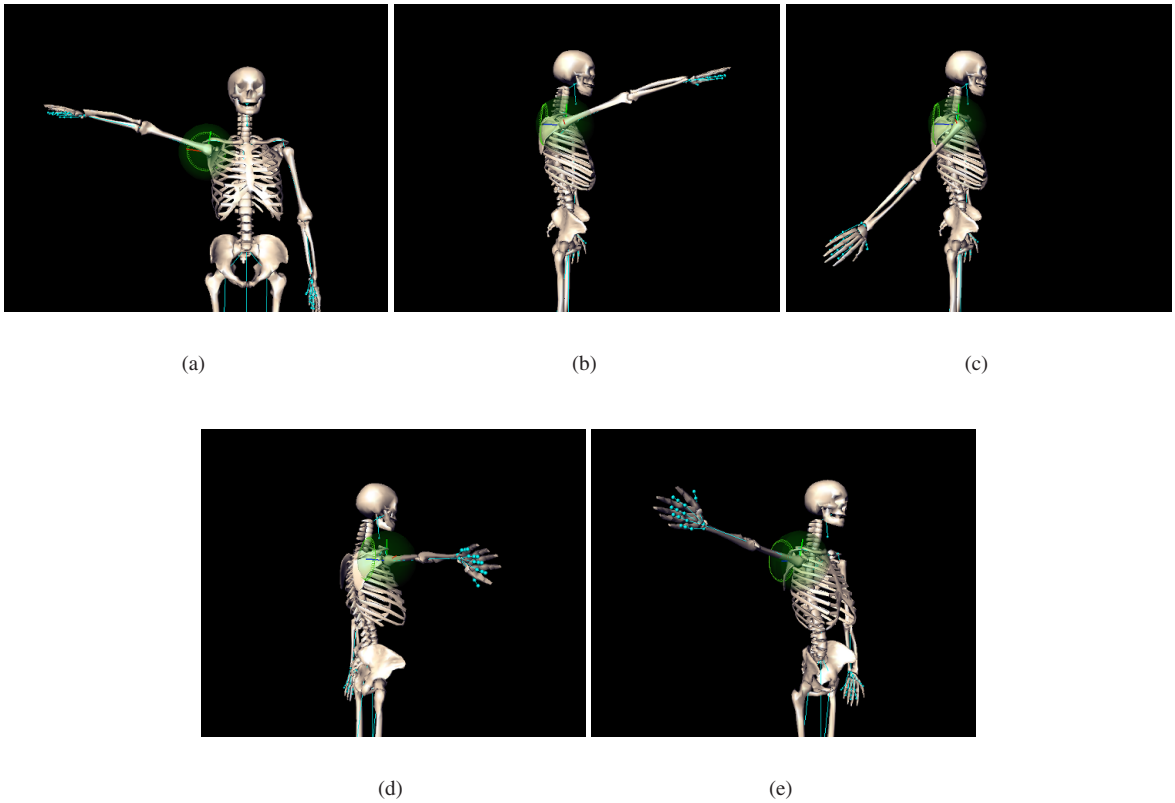
Figure 6.68: Range of motion on each axis and in each direction, for the sterno-clavicular joint: (a) abduction; (b) adduction; (c) flexion; (d) extension.

able or even painful to reach and are therefore logically avoided by the motion capture subject. Furthermore, as in the range of motion assessments of Luttgens and Hamilton, these are measured while exerting a pressure on the limb to push it to its rotational limit. In our case, we have recorded only the active range of motion, as opposed to the active **plus** passive range that follows from pressure exertion.

6.4.3 Inter-subject variance

To illustrate the relative insensitivity of these measurements across subjects, we have gathered motion data for two additional people, one of each sex. In Figure 6.71, we overlay the sets of quaternions for each additional person on those corresponding to the reference subject. Visual inspection in 3D shows that they superpose well. This is confirmed by computing the average closest-point distance between the points of the data sets to be compared, as well as the corresponding standard deviation. The computed values demonstrate the similarity between the measures over the entire range of motion.

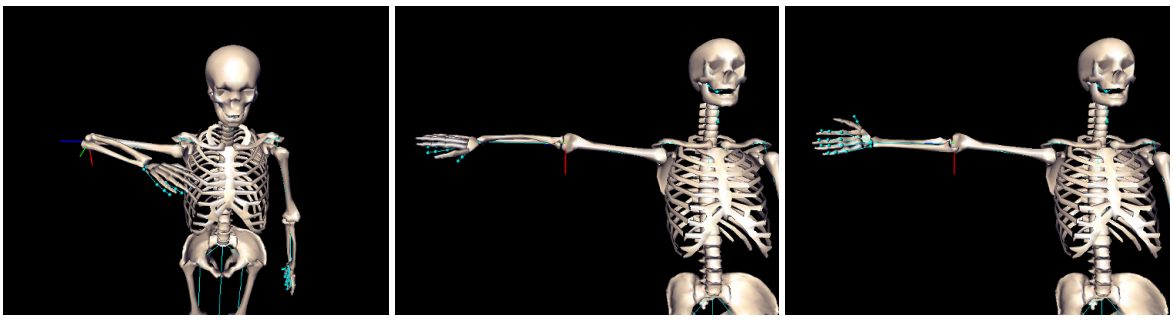
In the online Resources for the Humanoid Animation Working Group [Consortium1997], the difference in range of motion of women over men is given, and an excerpt of this data for the shoulder and arm is shown in



(a) (b) (c)

(d) (e)

Figure 6.69: Range of motion on each axis and in each direction, for the gleno-humeral joint: (a) abduction; (b) flexion*; (c) extension*; (d) inner axial rotation (thumb down); (e) outer axial rotation (thumb up).

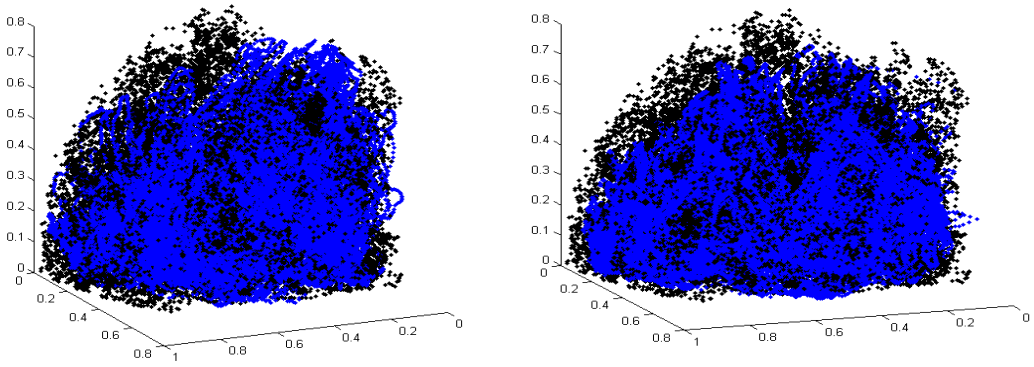


(a) (b) (c)

Figure 6.70: Range of motion on each axis and in each direction, for the elbow joint: (a) flexion; (b) inner axial rotation (thumb down); (c) outer axial rotation (thumb up).

Table 6.7 for the motion components where a discrepancy can be observed.

The variance becomes larger as one moved towards the limb extremity, but is relatively small at the elbow level and minimal at the shoulder level. Men may have greater physical strength, but the fair sex can find consolation in their superiority in terms of joint mobility. We therefore recommend that measurement of range



(a) (b)

Figure 6.71: Comparing subjects against each other. In black, the data for the female reference subject. In grey, the data corrresponding to a second subject. For each two sets, we computed the average distance in terms of closest points, as well as the standard deviation. (a) Measures corresponding to a second female subject, for which the average closest point distance is 0.0403 and the standard deviation 0.0500. (b) Measures for a male subject, with average distance 0.0314 and standard deviation 0.0432.

Movement	**Difference (degrees)**
Wrist flexion/extension	14
Wrist abduction/adduction	11
Elbow flexion/extension	8
Shoulder abduction (rearward)	2

Table 6.7: Average increase in range of joint movement of women over men.

of motion be carried out on female subjects, at least for the joints where variance is high, so as not to end up with too restricted joint limits.

In terms of inter-individual variance without sexual discrimination, the investigations of [Wang *et al.*1998] seem to indicate that such variance is minimal for young and healthy subjects, therefore confirming that no measurements over a large group of individuals is necessary to derive meaningful joint limits.

In this chapter, we have laid out our methodology for measuring joint range of motion and deriving a joint limits representation from the collected data. To show that our representation does allow to capture the real motion boundaries, we have compared it to data found in the literature, and thus found that it corresponds, give or take a few degrees. As we have said before, our point was not to build a database of range of motion data, but rather to propose a means of representing it, and we have successfully done so, not only for single joints, but also for coupled joints. This closes the theoretical chapters of the present work, leaving only various relevant implementation details, which are discussed in the following chapter.

Chapter 7

Implementation

In this chapter, we will review some relevant details about the implementation of the various software modules developed in the context of this thesis. The first section describes the body model used and the design of the various joint limit types that were implemented. The next section deals with the application developed for inspecting and verifying the conversion from the measured range of motion data to quaternions. The third section details the optimisation method applied to carry out body tracking, and how constraint enforcement was performed and integrated into the optimiser. Last of all, we will quickly present the visualisation tools used to test the joint limits both in the context of character animation and posture recovery from video sequences.

7.1 Human body model and joint limits constraints

The articulated structure representing the body model skeleton is implemented in the form of a generic Scene Graph, in other words a hierarchy of nodes of two types: *transform nodes* or *object nodes*. Transform nodes implement all transformations relative to their child object node(s), i.e. the transformations that situate the child

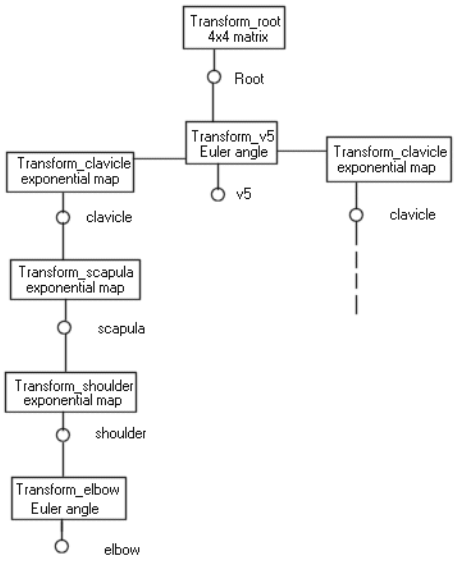


Figure 7.1: Scene graph hierarchy.

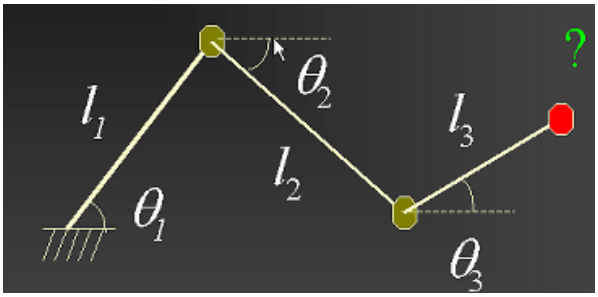


Figure 7.2: Articulated 2D structure with three joints and three segments. The transformation at joint 1 consists of a rotation θ_1 with respect to the horizontal plane, which orients the segment, followed by a translation of l_1 along the segment vector. This then situates joint 2 in space. Combining these transformations, we can determine the position of the end effector (the last joint in the structure).

object in the scene. An example of such a graph is shown in Figure 7.1

As the objects of our graph are exclusively of the joint type, given that we dealing with articulated structures and not with actual computer graphics scenes, our object nodes are from here on referred to as *joint nodes*. An articulated structure is a succession of nodes such as those depicted in Figure 7.2.

The transform nodes can vary in type from one node to another, as we did not wish to be constrained by a certain rotation paradigm. The supported types are:

- axis angles,
- Euler angles,
- exponential maps,
- 3x3 matrices,
- 4x4 matrices,
- quaternions.

To make our model standard and portable, its joint hierarchy can be converted so as to be H-Anim compliant. This feature furthermore ensures that the local joint co-ordinate systems are always likewise oriented. The necessity for this arises namely in the case of the enforcement of joint limits, as a range of motion is always defined with respect to a certain axis configuration. To illustrate this, take the example of a joint such as the elbow, where abduction is not possible. If we initially decide that abduction is rotation around the x-axis, then we need to make sure that the x-axis orientation always corresponds to an abduction motion (see 7.3). In the absence of such axis alignment, the local referential would be oriented according to the rotation occurring at the previous joint in the hierarchy, which can be arbitrary.

The limbs of the skeleton are the segments connecting one joint node to another, their length being defined by a 3D translation $(0., 0., T)$ multiplied by a translation coefficient c_T. The width of a limb is likewise defined by a width W multiplied by a width coefficient c_W.

Shape is added to our articulated structure by a variety of shape primitives:

- spherical metaballs (balls), defined by a 3D translation (t_x, t_y, t_z) with respect to its parent joint, and a scaling factor s;
- infinite cylinders, defined by a 3D translation (t_x, t_y, t_z) and a scaling factor s;
- ellipsoidal super-quadrics, defined by a 3D translation (t_x, t_y, t_z), a 3D rotation (r_x, r_y, r_z), and scaling factors along all three axes (s_x, s_y, s_z);

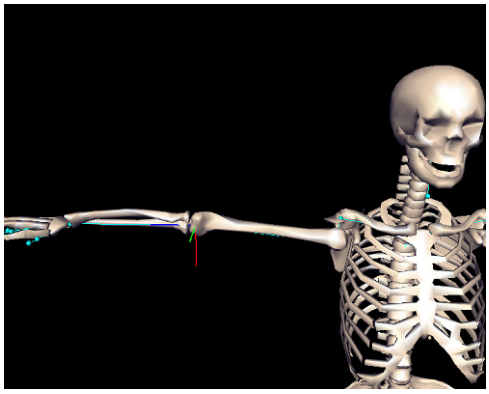


Figure 7.3: Example of an elbow referential where the x-axis is oriented towards the front of the body, the y-axis downwards, and the z-axis along the limb. Abduction would in this case be a rotation around the x-axis, and if a transformation at the shoulder level modified the referential at the elbow, we would need to rectify the referential so that it is locally always oriented in the same manner.

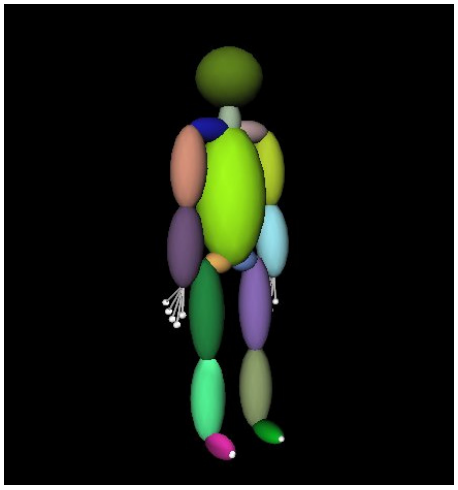


Figure 7.4: Visualisation of the body model.

- soft objects, defined by the same parameters as the ellipsoidal super-quadric. The difference with respect to the previous primitive resides in its surface definition, as we will see later.

The first three shape primitives are all sub-sets of the last type. An example of a body model fleshed out with soft objects is shown in Figure 7.4.

The body model so defined can be subject to various constraints on all its optimisable parameters. A limit can be set on the contraction/expansion factor of the limb lengths and widths. As to the rotational parameters, they can be limited either by implicit surfaces, min-max limits on Euler angles, or spherical polygons. The latter two limits can be defined either by the user, who sets the range of values or loads the vertices of a spherical polygon. Alternatively, they can also be computed from implicit surface joint limits. Conversion to the min-max representation is easily done simply by computing the two intersection points of the implicit surface with each co-ordinate system axis, as was done in Section 6.4.2.

An implicit surface is converted to its equivalent spherical polygon by first uniformly tessellation a sphere centred in (0., 0., 0.), using the algorithm of [Saff and Kuijlaars1997]. Each tessellation point corresponds to an upper arm orientation that we now need to test for validity. We convert this orientation into a unit quaternion, and then use the vector part of the quaternion (q_x, q_y, q_z) to determine whether it belongs or not within the implicit

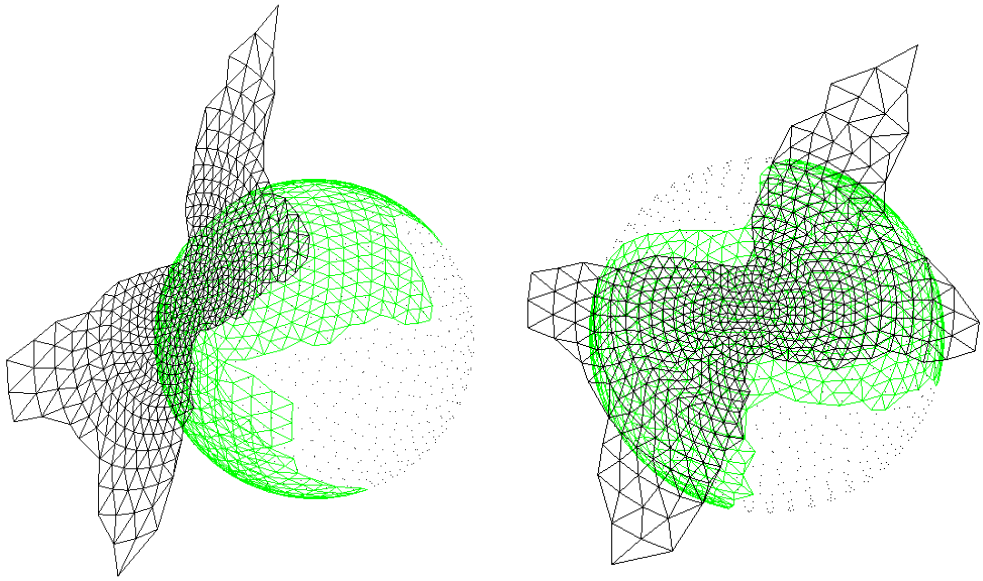


Figure 7.5: Spherical polygon derived from sphere tessellations, with the 2D planar projected triangulation displayed.

surface by evaluating the density function of the surface. If the result is higher or equal to the surface iso-value, then the rotation is valid.

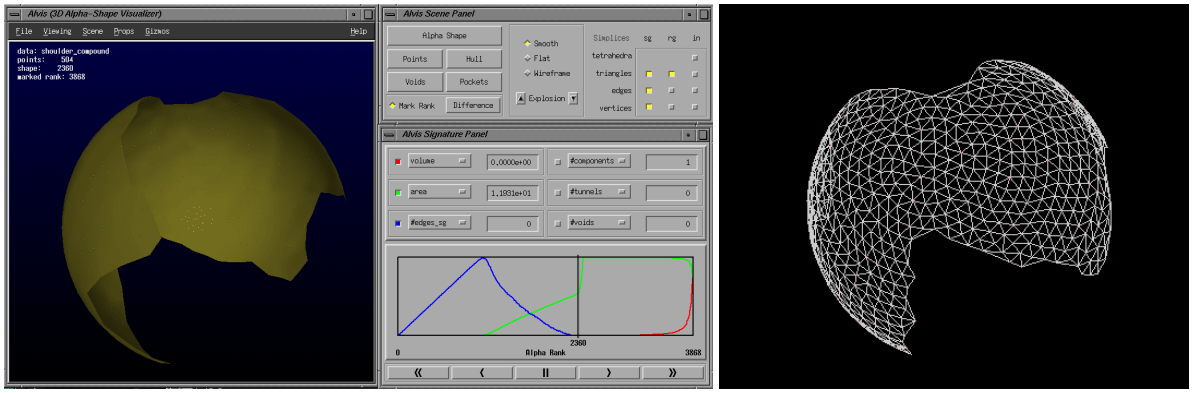
All tessellation points that pass this test are kept as "inside" points of the spherical polygon we are trying to compute. For this, we perform a Delaunay triangulation on these points. We need to take into account the fact that our polygon is not convex in general, but that a Delaunay triangulation always is, by definition. Therefore, we need to eliminate the boundary edges of the triangulation that do not belong to the polygon. For this, we first flatten our representation, so we perform a planar projection of the spherical Delaunay triangulation thus obtaining a 2D Delaunay triangulation, as shown in Figure 7.5. We simply take the middle point of each edge, and test whether it is a valid orientation. If the point does not belong, then the edge is removed. Once we have a consistent triangulation, we determine the convex hull of the points we kept, i.e. the smallest convex set of points that includes the valid points [Watson1992]. This gives us the border edges of the spherical polygon, which we re-order and connect, and then finally re-project onto the sphere in order to obtain the final spherical polygon.

An alternative method for converting an implicit surface to a spherical polygon was offered to us when we discovered the existence of alpha-shapes. By using all the valid tessellation points computed as explained above, we can directly process this data using the Alvis application [Edelsbrunner and Mücke1994]. In this manner, we obtain the connected triangulation border directly, this forming the spherical polygon. The result of such a conversion of the shoulder compound data is shown in Figure 7.6.

7.2 Range of motion measurement

Various functionalities have been implemented in order to enable us to verify the correctness of our motion capture to quaternion data conversion. This module uses a body model as defined in the previous sub-section, and on the basis of the captured 3D marker trajectories converted into quaternion rotations, we can apply the latter to the model and visualise the resulting animation. If the output animation is identical to the motion performed during the data collection session, then we can safely conclude that our conversion is corrected.

In Figure 7.7, we show such a derived sequence of quaternions rotations being applied to a model, allowing a visual comparison of the result with the original motion capture session. This module furthermore allows the loading of virtual markers, placed at the some locations as during the real motion capture session. The model can



(a) (b)

Figure 7.6: Spherical polygon derived from the alpha-complex of the valid tessellation points: (a) smooth surface in Alvis; (b) triangulated spherical polygon.

be interactively animated and the trajectories of the virtual markers processed in the same manner as the motion capture data. The resulting quaternions from the real and virtual sessions can then be compared to ensure the correctness of our conversion functions.

7.3 Posture optimisation and constraint enforcement

7.3.1 Posture recovery

In order to derive the posture of the body model from the stereo data, we apply a least-squares optimiser, whose principle is described in the Appendix. The body model being defined as an articulated structure with volumetric primitives defining its shape, we will be minimising the distance $F(x_i, \Theta)$ between the n 3D stereo points x_i and the skin surface corresponding to the sum of the field functions of the primitive(s). $\Theta = (\theta_1, ..., \theta_m)$ corresponds to the vector of joint angle values defining the current posture of the model.

In the absence of constraints, fitting the model to n stereo data points x_i simply amounts to minimising:

$$\sum_{x_i \in Data} F(x_i, \Theta)^2$$

with respect to Θ. In order to eliminate observation points that are too remote with respect to a given body part, a radius of influence is defined for all shape primitives, and points that are at a distance beyond this threshold are considered as outliers. These outliers are then given a least-squares optimisation weight of zero, whereas all inliers are attributed a weight of 1.0. As outliers have a non-negligible influence on least-squares optimisation, outlier elimination is absolutely vital in the case of noisy 3D data.

To determine the minima of F, we compute the minima of its derivative. For this we compute the Jacobians with respect to a given parameter $\theta_j \in \Theta$:

$$\frac{\partial F(x, \Theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} F(x_i, \Theta)$$

The quadratic distance from a 3D point x_i to a body part is defined as:

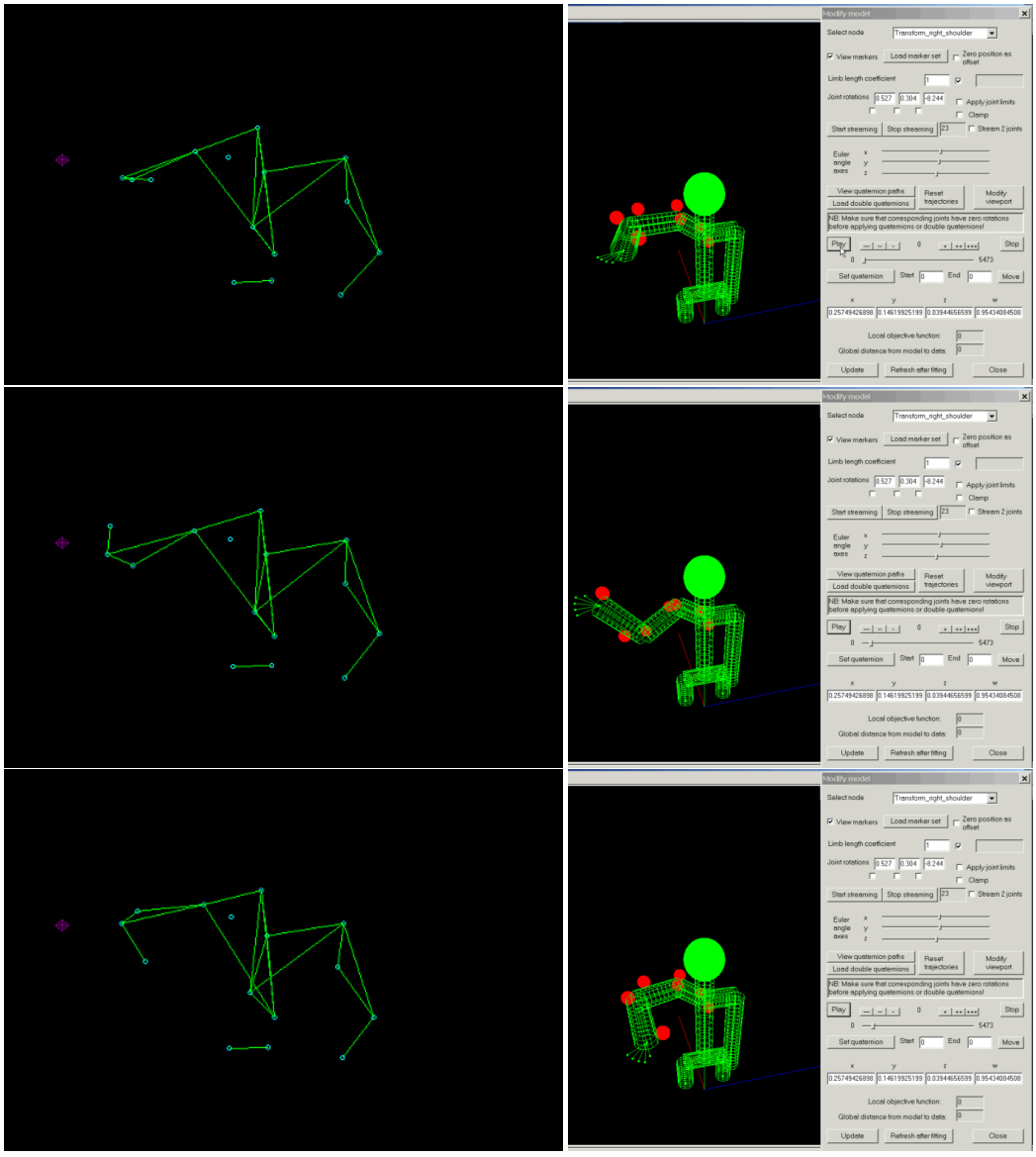


Figure 7.7: Visual comparison of the original motion capture session for the sterno-clavicular and gleno-humeral joints, visualised in Vicon Workstation^{TM} and the derived rotations applied to a simple articulated structure in our application.

$$F(x_i, \Theta) = x_i^T . Q_\Theta^T . Q_\Theta . x_i \tag{7.1}$$

where Q_Θ is:

$$Q_\Theta = L_\Theta . C_\Theta . S_\Theta$$

L_Θ is the local scaling matrix of the primitive:

$$L_\Theta = \begin{bmatrix} \frac{1}{c_W l_x} & 0 & 0 & 0 \\ 0 & \frac{1}{c_W l_y} & 0 & 0 \\ 0 & 0 & \frac{1}{c_T l_z} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $L = (l_x, l_y, l_z)$ are the radii of the soft object in each direction, defined as:

$$\begin{aligned} l_x &= W.c_w.s_x \\ l_y &= W.c_w.s_y \\ l_z &= T.c_T.s_z \end{aligned}$$

C_Θ is the local translation matrix:

$$C_\Theta = \begin{bmatrix} 1 & 0 & 0 & -c_w t_x \\ 0 & 1 & 0 & -c_w t_y \\ 0 & 0 & 1 & -c_t.t_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and S_Θ the multiplication of all the 4×4 homogeneous matrices from the root node of the articulated structure up to the transform node to which the shape primitive is attached. This formalism was outlined in [Plänkers and Fua2001], and the expression of the derivatives with respect to translation, rotation, width and shape primitive parameters are directly derived from eq.(7.1) and can be looked up in the same publication. The general expression of the derivative of the distance function with respect to a parameter θ_j is:

$$\frac{\partial F(x_i, \Theta)}{\partial \theta_j} = 2.x_i^T.Q_\Theta^T.\left[\frac{\partial Q_\Theta}{\partial \theta_j}\right].x_i$$

Least-squares optimisation is then carried out in series of iterations, this feature having been added so that the process does not get stuck in local minima. For a more detailed explanation, see the Appendix.

7.3.2 Constrained least-squares

In order to apply various constraints to the least-squares optimisation process, we use a task-priority strategy enforced by damped least-squares, the latter being explained in the Appendix. In such a formulation, the tasks are ordered from highest to lowest priority, and the error for each task is minimised while higher priority tasks remain satisfied at all times. The technique is popular in robotics applications, where it offers an attractive solution to the carrying out of conflicting tasks [Baerlocher and Boulic1998], replacing the commonly-used weighting technique that resulted in compromise solutions where none of the goals were actually reached.

When enforcing joint limit constraints, minimising $F(x_i, \Theta)$ becomes the lowest priority task. The Jacobian of F is written J_F. Let us assume that we are given a constraint C, with highest priority and Jacobian J_C. Our problem then becomes minimising $F(x_{i,}\Theta)$ subject to $C(\Theta) = 0$.

The least-squares framework iteratively increments Θ until the best solution is found that solves

$$x_i = J_F.\Delta\Theta \tag{7.2}$$

The pseudo-inverse J_F^+ of the Jacobian matrix gives an approximation of the increment $\Delta\Theta$ as

$$\Delta\Theta = J_F^+ x_i \tag{7.3}$$

thus involving iteratively adding to Θ increments proportional to:

$$\Delta\Theta_0 = J_F^+ \left[F(x_1, \Theta), ..., F(x_m, \Theta)\right]^T \tag{7.4}$$

If we solve for the task with highest priority, namely constraint C, we write:

$$J_C.\Theta_z = 0 \tag{7.5}$$

Θ_z being solution to the equation and, by analogy with eq.(7.3), can be expressed as:

$$\Theta_z = J_C^+ C(\Theta) \tag{7.6}$$

$(I - J_C^+ J_C)$ projects onto null-space, meaning that we have:

$$J_C(I - J_C^+ J_C)\Theta_z = 0 \tag{7.7}$$

Combining eq.(7.6) and eq.(7.7), we get:

$$\Delta\Theta = J_C^+ C(\Theta) + (I - J_C^+ J_C)\Theta_z \tag{7.8}$$

We solve this and substitute the resulting solution vectors $\Delta\Theta$ in the equation of the lower priority constraint, eq.(7.3). The resulting equation is solved for Θ_z using the pseudo-inverse matrix which is then substituted back, giving the solution for F that satisfies the constraint C:

$$\Delta\Theta = J_C^+ C(\Theta) + [J_F(I - J_C^+ J_C)]^+ (x_i - J_F(J_C^+ C(\Theta))) \tag{7.9}$$

We illustrate such a damped constrained least-squares optimisation with the simple example of a short articulated structure. The optimisation results are show in in Figures 7.8 and 7.9. In this case, two hierarchical 3D implicit surface were used to constrain the first two joints of an articulated structure, composed of three rotational joints and three segments. The objective function to be minimised is the distance from the end effector to the 3D data. The constraint on the first joint has priority over the constraint on the second joint.

In summary, all that is needed to enforce constraints is the ability to compute their Jacobian with respect to state variables. How these Jacobians are defined will be detailed in the upcoming sub-section.

In certain cases, we can nevertheless happen upon a singularity when the various tasks have conflicting goals as in the case of Figure 7.10. In the depicted configuration, the first joint has a motion range limited to $90\,\mathrm{deg}$ so that the segment reaches its boundary position when placed vertically. The second joint has full mobility, but no solution exists where the end effector reaches its goal and the first joint is within its motion range. When attempting to reach the target, the first joint might will remain at its limit value (Figure 7.10(b)), one dimension thus being lost in state-space. The pseudo-inverse solution in this case tends to maximise the contribution of the limited joint, this yielding the solution shown in Figure 7.10(c), which in terms of combined task error is not optimal. Solutions exist for circumventing such singularities, for example in [Boulic *et al.*1997].

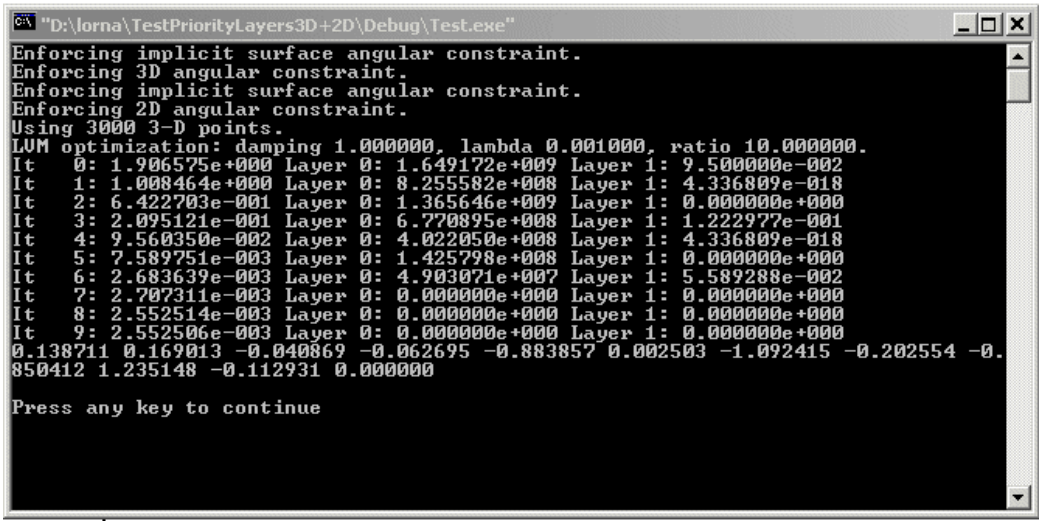
7.3.3 Constraint derivatives

7.3.3.1 Implicit surface joint limits

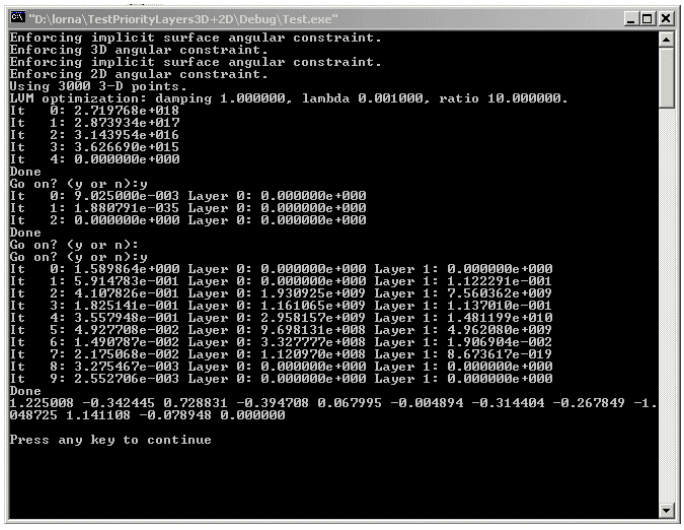
The implicit surface formulation lets us enforce constraints in a very simply manner:

1. For the parent joint, determine whether its rotation is valid by evaluating its density, and its derivatives with respect to joint angles if it turns to be valid or not.

2. For the child joint, determine to which voxel its parent rotation belongs, load the corresponding child joint limits, and verify its validity. In the case of an invalid rotation, evaluate the derivatives using the corresponding implicit surface representation. This allows us to express a lower priority constraint using the corresponding field function.



(a)



(b)

Figure 7.8: Fitting of an articulated structure to 3D data: (a) In this case, all layers were optimised simultaneously; (b) here, task-priority least-squares fitting was applied, where a first solution is found, where the constraints are satisfied, and further optimisation is then carried out to minimise the distance from the end effector to the 3D data.

Given a state variable θ defined as a rotation $\theta = (\theta_x, \theta_y, \theta_z)$[1], then the constraining objective function is the square of the total density function of the implicit surface. The total density function is defined as:

[1]The rotation corresponds to the nature of the data enclosed within the implicit surface. If the implicit surface envelopes 3 DOF rotations expressed as unit quaternions, then θ is a unit quaternion. If the surface encloses 2 DOF rotations expressed as Euler angles, then θ is a Euler angle.

Figure 7.9: Least-squares fitting of the articulated structure to 3D data, using the implicit surface constraint. As we can see, in the first half of the steps, the end effector does not move closer to the data, as the constraint layer is being optimised. Once these constraints satisfied, the distance from end effector to data is finally minimised.

$$D(\theta) = \sum_{k=1}^{p} f_k(\theta) \tag{7.10}$$

where f_k is the field function of one of the implicit surface's p primitives, of centre $(x_{c_k}, y_{c_k}, z_{c_k})$. Given that the surface is defined as all points in 3D space where $D(\theta)$ is equal to the iso-value iso, the objective function is:

$$f_{obj} = (D(\theta) - iso)^2 \tag{7.11}$$

It should be pointed out that unlike exponential-function based primitives such as those in [Plänkers2001], spherical primitives with the field function defined as in eq.(7.12) have a more continuous behaviour for $D(\theta)$ and f_{objf}. We illustrate this phenomenon in Figure 7.11, for the case of two primitives, where we can see that the total density function has more than one peak, the ensuing objective function therefore having several local minima.

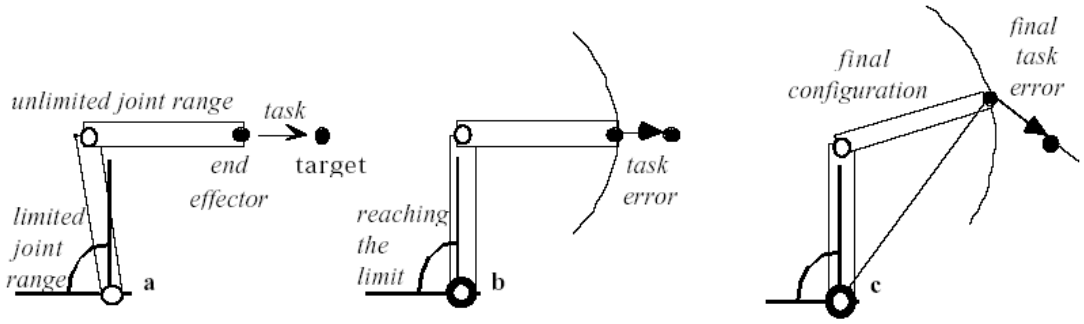
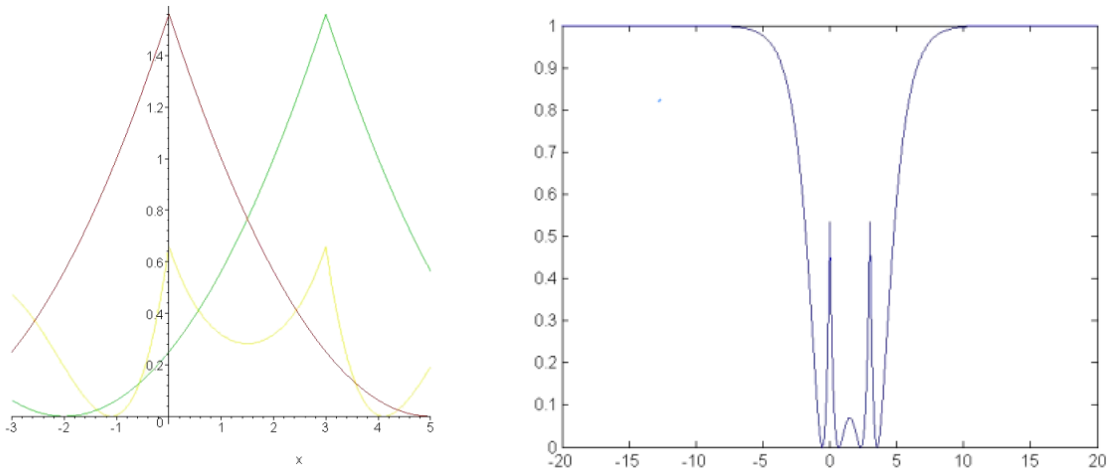


Figure 7.10: Conflicting task goals: (a) the end effector is to reach the target point, the first joint have a limited range of motion and the second not; (b) during optimisation, the first joint reaches its motion boundary; (c) the final solution maximises the contribution of the limited first joint [Boulic *et al.*1997].



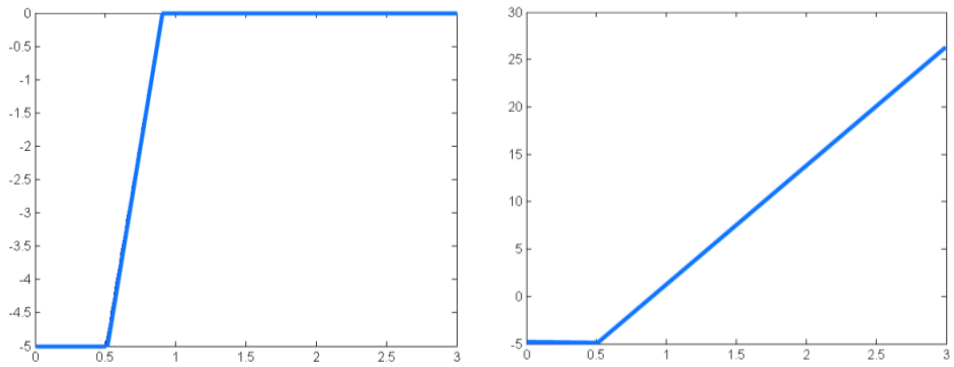
(a) (b)

Figure 7.11: Field functions and objective function for two primitives in the 1D case, where the iso-value is 1.0, thickness e_i is 0.5 and stiffness k_i is 1.0. The first primitive is located at $x_{c_i} = 0.0$ and the second primitive at $x_{c_i} = 3.0$: (a) In darker colours, we have the field function corresponding to the two primitives, respectively. As can be clearly observed, the total density function for the two primitives has at least two peaks; (b) the resulting objective function for these two primitives, as per eq.(7.11) showing four local minima.

The density function of the spherical primitive has been modified from eq.(5.5) so as not to have a finite radius of influence. The reason for this is that the density would otherwise be equal to zero at very distant points and would yield an objective function with several local minima, as in the case of an exponential field function. The field function has therefore been modified to:

$$f_i(\theta) = \begin{cases} -k_i r + k_i e_i + 1 & \text{if } r \in [0, e_i] \\ \\ \frac{1}{4}\left[k_i(r - e_i) - 2\right]^2 & \text{elsewhere} \end{cases} \tag{7.12}$$

where r is the Euclidean distance from θ to the primitive's centre. In Figure 7.14, we show the field function, with and without radius of influence.



(a) (b)

Figure 7.12: (a) The spherical primitive field function with a cut-off value at the radius of influence; (b) the same field function defined everywhere.

In terms of constraints, the objective function is defined as:

$$f_{obj}(\theta) = \begin{cases} (D(\theta) - iso)^2 & \text{if } D(\theta) < iso \\ \\ 0 & \text{elsewhere} \end{cases}$$

The gradients are directly derived from the expression of the objective function and are defined as follows:

$$\frac{\partial f_i(\theta)}{\partial x_{c_i}} = 2.(D(\theta) - iso).\frac{\partial f_i(\theta)}{\partial R_i}.\frac{(\theta_x - x_{c_i})}{r}, \quad \frac{\partial f_i(\theta)}{\partial y c_i} = 2.(D(\theta) - iso).\frac{\partial f_i(\theta)}{\partial R_i}.\frac{(\theta_y - y_{c_i})}{r},$$

$$\frac{\partial f_i(\theta)}{\partial z c_i} = 2.(D(\theta) - iso).\frac{\partial f_i(\theta)}{\partial R_i}.\frac{(\theta_z - z_{c_i})}{r} \tag{7.13}$$

where

$$\frac{\partial f_i(\theta)}{\partial R_i} = \begin{cases} -k_i & \text{if } r \in [0, e_i] \\ \\ \frac{k_i^2}{2}(r - e_i) - k_i & \text{elsewhere} \end{cases}$$

The field functions, objective functions and gradients are plotted in Figure 7.13.

If the implicit surface envelopes quaternion joint rotations, and the rotational parameters of the model we are optimising are expressed in a different formalism, then the gradient matrix of eq.(7.13) needs to be multiplied by a partial derivative matrix, following the chain rule. For example, if we are optimising a Euler angle $\Theta = (e_x, e_y, e_z)$, and we wish to compute the gradient with respect to e, then this gradient would be defined as:

$$\frac{\partial f_{obj}}{\partial \Theta} = \frac{\partial f_{obj}}{\partial q}.\frac{\partial q}{\partial \Theta} \tag{7.14}$$

where $q = (q_x, q_y, q_z, q_w)$ is the equivalent unit quaternion of the Euler angle, obtained by the conversion formula given in the Appendix.

The first term of eq.(7.14) is given by eq.(7.13), and the second term, the Jacobian matrix, is computed as:

$$\frac{\partial q}{\partial \theta} = \begin{bmatrix} \frac{\partial q_x}{\partial e_x} & \frac{\partial q_y}{\partial e_x} & \frac{\partial q_z}{\partial e_x} \\ \frac{\partial q_x}{\partial e_y} & \frac{\partial q_y}{\partial e_y} & \frac{\partial q_z}{\partial e_y} \\ \frac{\partial q_x}{\partial e_z} & \frac{\partial q_y}{\partial e_z} & \frac{\partial q_z}{\partial e_z} \end{bmatrix} \tag{7.15}$$

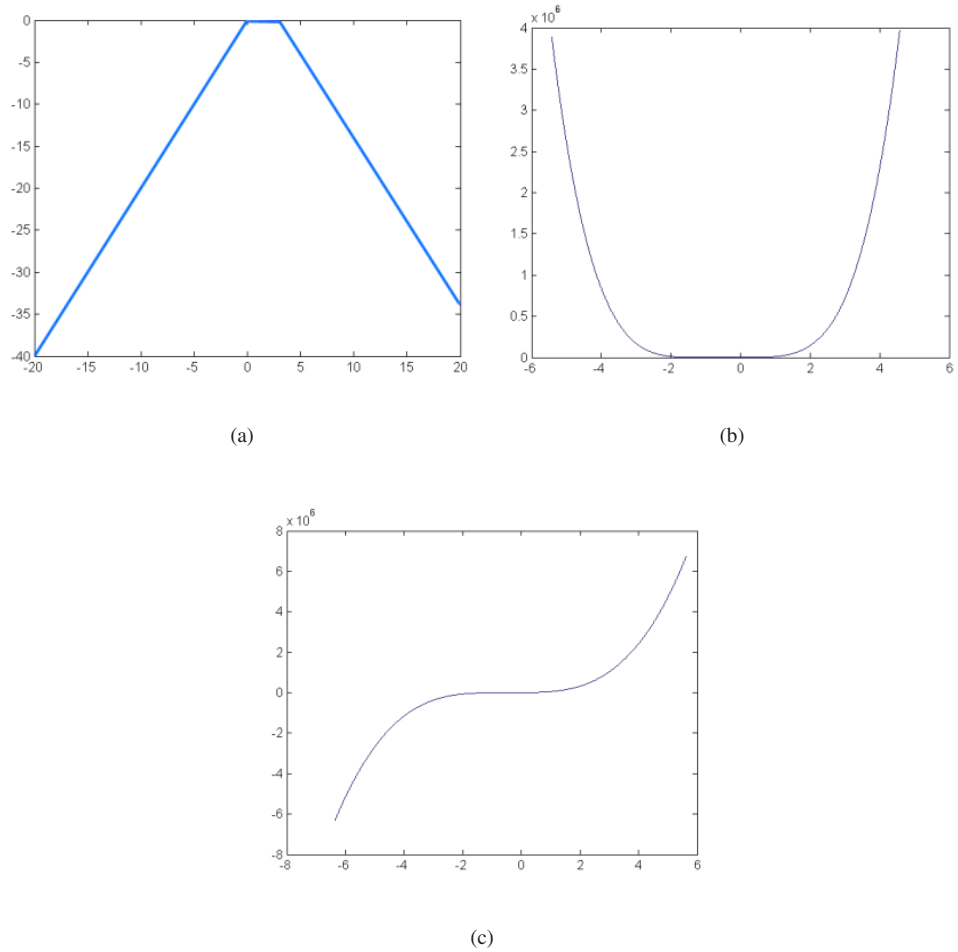


Figure 7.13: (a) Total density function using the field function of eq.(7.12) for the same two primitives as in Figure 7.11. We note that there are no longer several peaks; (b) Objective function value along a line drawn through the middle of an implicit surface with 16 primitives; (c) Gradient along the same line.

Let us now consider the reverse case, where the implicit surface envelopes Euler angles, and we are optimising a quaternion rotation $\Theta = (q_x, q_y, q_z)$ in our body model. The gradient with respect to q would be defined as:

$$\frac{\partial f_{obj}}{\partial \Theta} = \frac{\partial f_{obj}}{\partial e} \cdot \frac{\partial e}{\partial \Theta} \tag{7.16}$$

where $e = (e_x, e_y, e_z)$ is the equivalent Euler angle of the quaternion, obtained by the conversion formula given in the Appendix. The first term of eq.(7.16) is given by eq.(7.13), and the second term, the Jacobian matrix, is computed as $\frac{\partial e}{\partial q}$, by analogy with eq.(7.15).

In all other cases, where the rotation paradigm of the model parameters is identical to the rotations limited by the implicit surface, the gradients are simply equal to eq.(7.13).

Impact of distance metric approximations We herewith remind the reader that in both cases, whether our implicit surfaces encloses unit quaternions or Euler angles, the distance metric we use is an approximation of the real distance. In both cases, we use the Euclidean distance for representing the distance between two rotations.

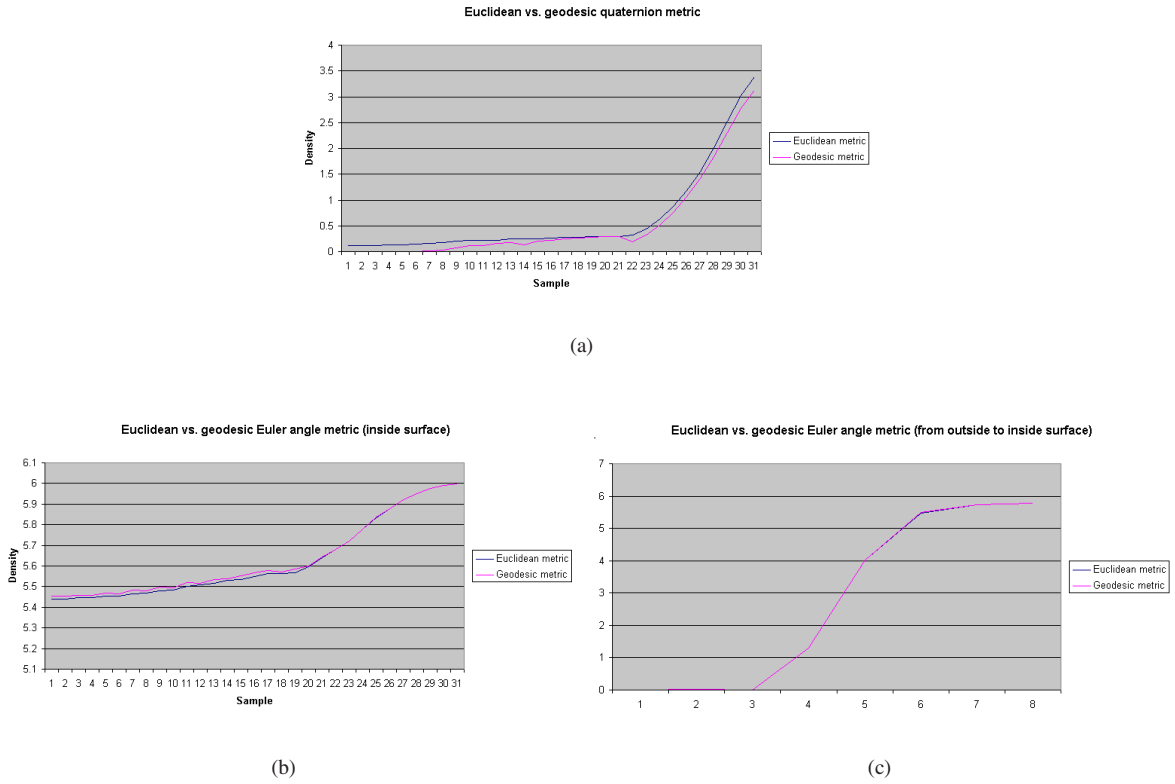


Figure 7.14: Assessment of the impact of using an approximation of the real metric. (a) Various rotations were generated by regularly incrementing the amount of one of the rotation components, the other two being kept constant, but non-zero. The implicit surface is composed of 20 primitives and is defined with an iso-value of 1.0. The density with respect to this surface is evaluated for each generated quaternion rotation, respectively with the real distance metric and with the Euclidean distance approximation. The rotations gradually move from outside the surface to inside it. (b) Rotations are generated by incrementing one rotation component, a second one being kept constant but non-zero, the remaining one being zero, as we are dealing with 2D Euler angles. The implicit surface is constituted of 60 primitives, with an iso-value of 5.0. The Euler angle rotations range from values outside the surface to values inside the surface, the progression within the surface being shown in (c).

In unit quaternion space, the real metric corresponds to the geodesic metric on the surface of the 4D sphere. For two quaternions p and q, this distance ρ is:

$$\rho(q,p) = 2.|\arccos(S(p^*q))|$$

where $S(q)$ is the scalar component of q, and $p^*$is the quaternion conjugate of p.

For two Euler angles, the real metric corresponds to the angle of rotation between the two, which can be obtained by computing the axis-angle rotation between the two resulting Euler angle orientations, its angle representing the real metric.

In Figure 7.14, we show the computed density values with respect to our implicit surface joint limits both in the case of quaternions and Euler angles. Whether comparing rotations inside or outside the surface, we note that using the real metric or an approximation of it has hardly any influence on the resulting densities, therefore justified the use of the latter.

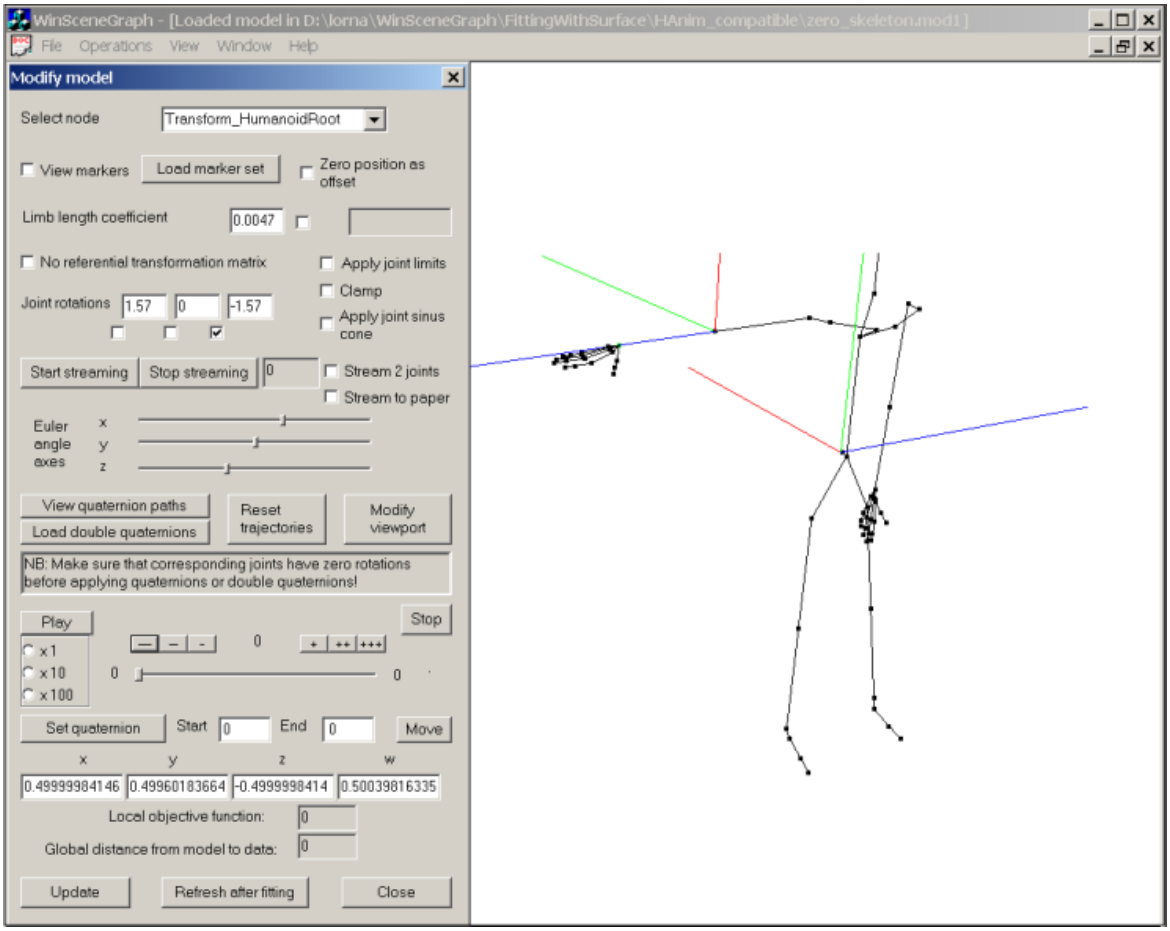


Figure 7.15: Graphical user interface displaying a human skeleton model and the various translational and rotational parameter values.

7.3.3.2 Spherical polygons and min-max values on Euler angles

For both these cases, the constraint has been implemented as a penalty function [Badler *et al.*1993], meaning that if the optimised parameters approach the boundaries of the min-max interval or spherical polygon, a large residual error value is returned.

7.4 Visualisation applications

Various visualisation applications were developed, each having its own specific functionalities. The first graphical user interface, developed on Windows^{TM} using the OpenGL graphics library, is shown in Figure 7.15 and allows to manipulate an articulated structure. All model parameters can be interactively modified, and least-squares optimisation run directly from the interface. The various parameter values can be inspected at each step, as well as the visual progress of the matching of the model to the data, and the evolution of the objective function value.

The application used to visualise the video tracking results was developed within our lab, on Windows^{TM} using the OpenInventor graphics package, and its interface is shown in Figure 7.16. In the top window is displayed the superposition of the metaball model with the 3D stereo data, and in the bottom window, the projection of the

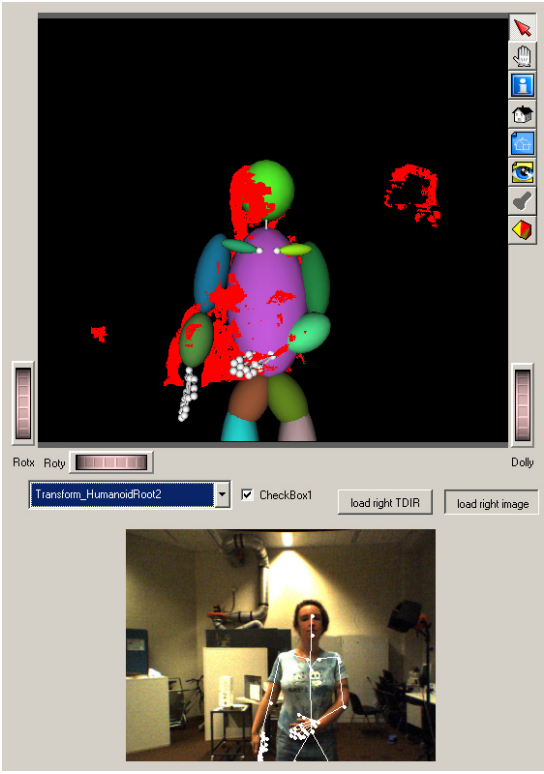


Figure 7.16: Visualisation of the structure and data in the OpenInventor GUI.

skeleton of our model onto the frames of the original video sequence.

Finally, we modified and adapted to our needs an animation application developed at the Virtual Reality Lab, on SGI Irix 6.5 and using OpenInventor [Aubel2002]. This software allows to display a multi-layered virtual human, composed of a skeleton, muscles, fat and skin. The user can then interactively modify the rotational values in order to visualise the resulting pose as well as the deformations corresponding to the various layers. Joint limits of various types are implemented, depending on the nature of the joints, and can be enabled or disabled at will, as well as be interactively designed. VRML keyframe sequences can be loaded and the resulting animation visualised. The standard look of the graphical user interface is shown in Figure 7.17.

Our contribution to this software was to implement an additional joint limits type, namely hierarchical or non- hierarchical implicit surfaces. We added the feature that allows for the current quaternion rotation of the joints to be tested, in terms of validity. This is carried out by simply computing the total density function for a given quaternion with respect to the implicit surface joint limits associated to the current joint, and comparing it to the iso-value of the surface. If the posture is not valid, then the nearest valid posture is computed.

The S^3 quaternion hyper-sphere containing the implicit surface is centred in the current joint, as shown in (Figure 7.18).

Hierarchical joint limits are applied in the following way:

- For a given posture of the parent joint, we test whether the rotation is valid with respect to the implicit surface joint limits of the joint; if it is not valid, we clamp it to the nearest valid rotation. Then we determine for the current posture in which voxel of the joint voxelisation it lies, and store the identifier *id* of the voxel. If the rotation does not lie directly within a voxel, we determine which voxel is the closest in terms of distance to its centre.

- For a given posture of the child joint, we test whether it is valid with respect to the implicit surface joint

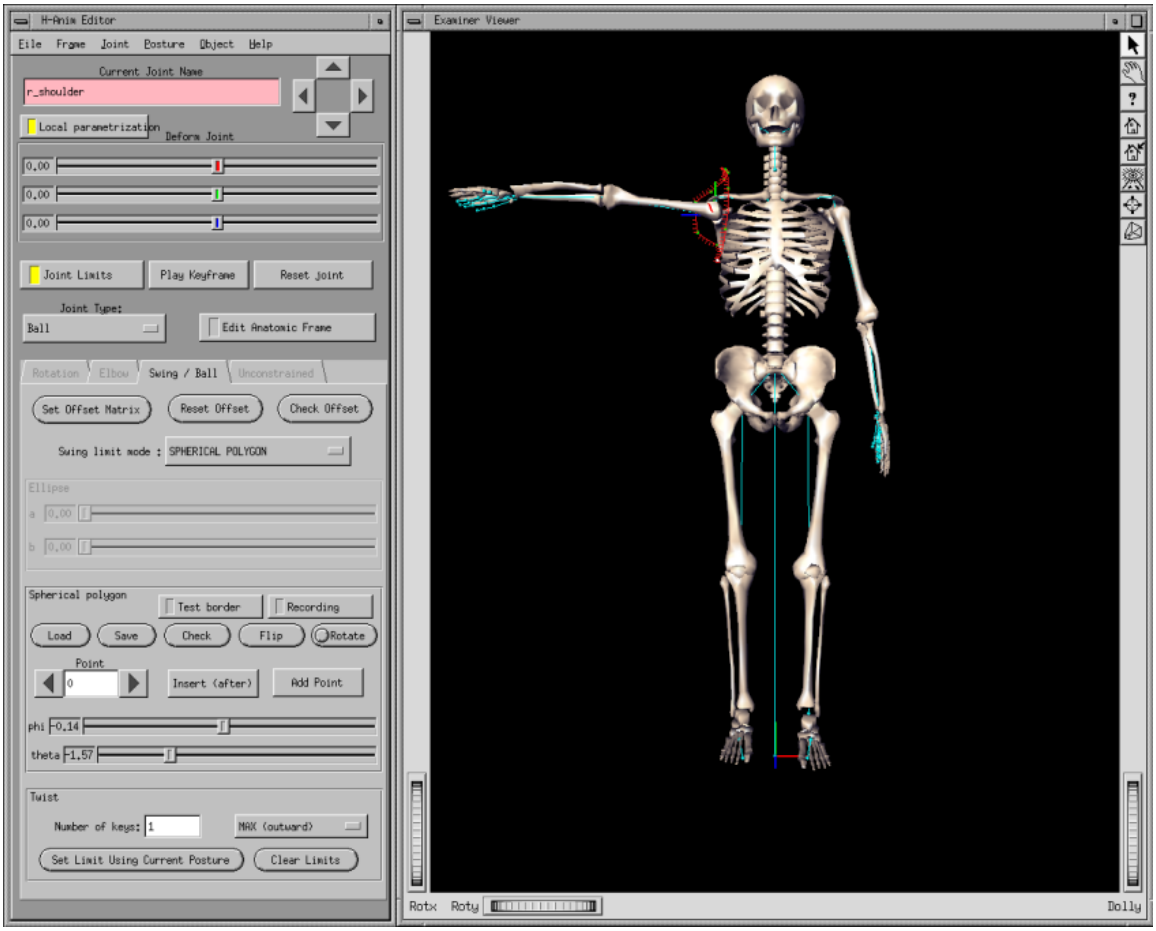


Figure 7.17: Graphical interface for animating a multi-layer humanoid, here with only the skeleton layer displayed.

limits corresponding to voxel *id*, these different joint limits per voxel being stored as a vector of implicit surfaces within the child joint.

In the case where a rotation is outside the implicit surface, and therefore not valid, we need to determine which is the closest valid rotation, and clamp the joint rotation to this value. For carrying out this task, we have implemented three separate schemes, depending on what information is available for the implicit surface joint limits:

1. In the first case, in the loaded model file, an implicit surface can be defined not only by its spherical primitives and its iso-value, but also by its surface vertices, obtained by the Marching Cubes algorithm. If we are confronted with an invalid rotation, we then simply determine which surface vertex is the closest in terms of Euclidean distance, and clamp the rotation to this surface vertex.

2. In the second case, we do not have the surface vertices, but are given the co-ordinates of the centre of mass m of these vertices. In this case, for an invalid rotation p, we perform dichotomy along the segment $[p, m]$, testing at each step whether the middle point is inside or outside the implicit surface, and sub-diving the left or right part of the segment in consequence. When the middle point is at the intersection of the current segment and the implicit surface, we have found our closest valid rotation.

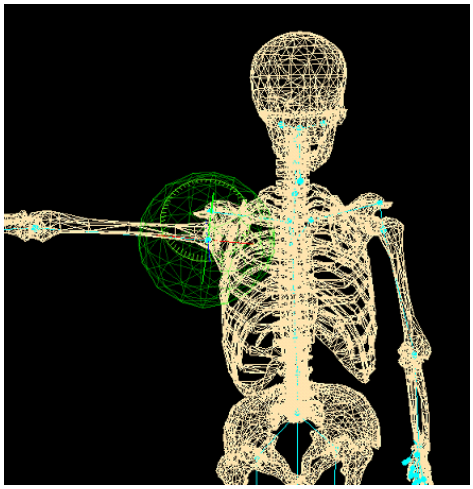
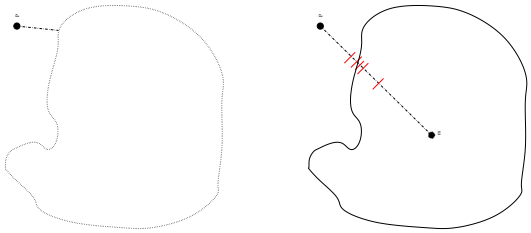


Figure 7.18: Sphere centred in joint, representing implicit surface joint limits.



(a) (b)

Figure 7.19: Clamping to closest valid rotation: (a) if p is the invalid rotation outside the implicit surface (shown in 2D), and if we know the surface vertices, we clamp to the closest vertex; (b) if we know only the centre of mass surface vertices, we proceed by dichotomy along the segment between p and m until we intersect the surface.

3. In the last case, neither of the above information is provided, so we simply move in the direction of the total gradient of the implicit surface, for the given invalid rotation p.

The first two of the above procedures are depicted in Figure 7.19.

In the case where joint limits are non-hierarchical, a given rotation is simply tested against the implicit surface joint limits of the joint, and clamped to its surface if the rotation is not valid.

Applying joint limits in such a manner allows us on the one hand to interactively visualise the boundaries of motion, and on the other, to load keyframe sequences with invalid rotations and obtain the resulting sequence with all invalid rotations clamped to their closest valid ones.

Chapter 8

Conclusion

8.1 Contributions

In this work, we have proposed to prune the solution state-space of motion capture applications by applying various types of constraints to motion capture frameworks.

8.1.1 Extrinsic constraints

Our first contribution consists in having applied extrinsic constraints to optical motion capture, that is, constraints that are based on information provided by other sources than the model. More specifically, we use knowledge of camera position and orientation, and limitations on frame-to-frame joint velocity and acceleration. Such constraints are almost omnipresent in any work in the motion capture context nowadays. 3D reconstruction, segmentation and posture recovery are all driven by such constraints, thus allowing not only the detection of errors in each module, but also their correction. Through such rigorous constraint enforcement, we have succeeded in driving the number of errors down to a strict minimum. The novelty of our approach does not lie within the definition of the constraints themselves, but in the manner in which they have been incorporated into the process. In this fashion, each individual step of the motion capture process not only becomes constrained, but also subject to a consistency constraint with respect to the steps that precede and follow it. These intra- and inter-task constraints can be summarised as follows:

1. 3D reconstruction is performed using epipolar geometry on two views, its accuracy being verified by coherence with the other camera views. Each reconstructed 3D marker position is verified in terms of visibility and occlusion in the camera views with respect to the current body model configuration.

2. The markers are tracked simultaneously in 2D and 3D, the condition of coherence between the two needing to be satisfied. Temporal constraints are applied to the frame-to-frame tracking of the markers to eliminate ambiguities. The marker segmentation result is verified against the current body model configuration, on the basis of the observed marker-to-joint distance. Additionally, markers attached to a same limb are bound to have rigid motion, this representing a further constraint on marker identification.

3. The body model posture is updated with respect to the reconstructed and segmented marker data.

8.1.2 Intrinsic constraints

Our second and main contribution resides in the context of intrinsic constraints, that is, constraints defined by the model itself. More specifically, we have advanced the state-of-the-art in terms of joint limits by proposing a new representation for such motion boundaries. Our representation is based on data collected from live subjects

and therefore reflects the real ranges of motion of a human being, contrary to existing joint limit schemes used in motion capture applications. Indeed, to this day in the computer vision domain, joint rotations have always been enforced by simple value ranges on Euler angle rotations, these being neither intuitive nor realistic. In computer graphics, more sophisticated joint limits have been investigated, but all apply, without exception, only to limb orientation, thus discarding the rotation of a limb around its own axis.

Our representation remedies all these short-comings, and exhibits the following strengths:

1. We chose the implicit surface representation to express our joint limits boundary as it has a concise mathematical expression, is smooth and continuous, nevertheless allowing detail in its shape, meaning that it can be locally influenced.

2. Implicit surfaces are expressed analytically, thus allowing immediate determination of the validity of a rotation, as well as of the rotation parameter gradients. Both are *sine qua non* conditions for constraining motion parameters in character animation and posture recovery in motion capture applications. In this manner, our representation is useful for computer vision as well as for computer graphics applications.

3. Data acquisition from live subjects using optical motion capture is simple and non-invasive, and allows the collection of a large number of samples within a small time frame.

4. The collected joint rotation data is converted to its corresponding quaternion rotations that can be represented as scattered 3D data if their unitary condition is enforced. This conversion is fast and uncomplicated and yields a compact data set to be approximated by an implicit surface.

5. Representing joint rotations as quaternions and computing an implicit surface envelope for them ensures that our joint limits are independent of the chosen rotation paradigm and that they include all motion components, meaning angular and axial rotation.

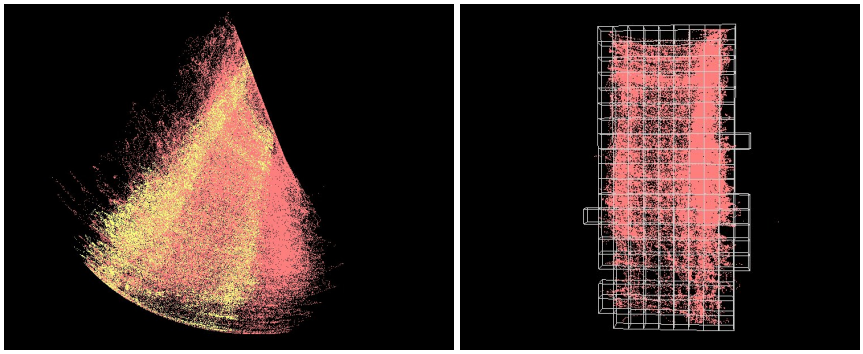
6. The process we use, from data collection to boundary extraction, is totally generic and can readily be applied to any joint of the human body model, insofar as its rotations are measurable.

7. The hierarchical joint limits scheme we propose allows to define motion boundaries for coupled joints, thus accounting for both intra- and inter-joint dependencies. Furthermore, this hierarchical model can be applied to a series of as many joints as necessary, the feasibility of this being only limited by processing power and by the amount of data available as a too high depth might lead to extreme data sparsity.

That each constraint category improves the performance of posture recovery has been demonstrated, by applying them separately to a motion capture application. The highest efficiency would be gained by applying both extrinsic and intrinsic constraints, so as to benefit from search-space pruning provided by each. We remind the reader that designing a constrained motion capture framework was not the objective of this work, our subject of focus being the constraints themselves and their usefulness in such a context.

8.2 Future research directions

Further work could be directed towards the computation of implicit surface joint limits, using however the real distance metric between rotations, namely the geodesic distance. In this case, unit quaternions could be used both for the 3 DOF and 2 DOF case. For the surface-like unit quaternions obtained for the latter, one could re-use the corresponding alpha-shape that was computed and shown in Section 6.2.1.3. Each triangle of the alpha-shape could then be approximated by a generalised metaball [Jin *et al.*2000], based on a triangular skeleton [Ilic and Fua2003]. In both the 3 DOF and 2 DO case, determining whether a rotation is valid or not, and computing the corresponding derivatives would be carried out in the same manner as in present work, except that the geodesic metric would be applied instead of the Euclidean distance. This being computationally heavier, processing time would inevitably increase, but in counterpart, one would gain in mathematical rigour. In the



(a) (b)

Figure 8.1: Comparison of range of motion and actual comfort zone: (a) quaternion data for the elbow, superposing the range of motion data with the comfort zone data; (b) voxelisation of the comfort zone Euler angle data.

case where the 2 DOF joint is the parent joint in a hierarchical coupling, the voxelisation method defining local rotation clusters would need to be replaced be a technique more suitable to the new shape of the data.

Some of the joint rotation data collected on several subjects high-lighted an interesting fact, namely that in normal conditions, the average human being has a natural tendency to avoid coming too close to the limits of range of motion. This is perfectly comprehensible in the sense that extreme joint rotations usually cause a feeling of discomfort, or even pain. This phenomenon is clearly visible in Figure 8.1(a), where we have superposed the data collected for assessing the complete range of motion of the elbow with the data classified as the "comfort zone" of the same joint.

Using the information provided by such data comparison, one could establish a probabilistic tracking scheme, where the rotations within the comfort zone have a higher likelihood than those outside. Such considerations could readily be integrated into a Multiple Hypothesis Tracking process, further pruning the state-space of solutions for posture recovery.

Another possible research direction could concern the nature of the collected motion trajectories. Frame-to-frame links being available, we could establish transition patterns, on the basis of voxelised data, such as the one shown in Figure 8.1(b). By analysing the trajectories through the data voxelisation, we can produce a transition graph, along with the transition probabilities from one voxel (or cluster) to another. Whether such transition graphs are meaningful for arbitrary motion remains to be seen, but in the case of a specific type of motion, such as walking, jumping or grasping, one could apply a probabilistic transition model to aid posture recovery.

Chapter 9

Appendix

9.1 Rotation representations

"Il ne faut pas multiplier à gauche, il ne faut pas multiplier à droite, il ne faut pas multiplier du tout, ça ne marche pas!"

Raquel Urtasun, July 2001

9.1.1 Matrices

A 3x3 matrix represents a rotation around one of the axes of the co-ordinate system, the rotation being defined by an angle θ. A rotation matrix is the multiplication of three such matrices, one for each axis of the co-ordinate system, depending on the order of rotation around the axes.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the rotation order is, for example, X, Y and Z, then we obtain:

$$R_z * R_y * R_x = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{9.1}$$

The 4x4 matrix is the result of the multiplication of a rotation matrix and a translation matrix, translation being either the last row or last column of the matrix, depending on the chosen convention.

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrices in general perform all types of transformations, these not being limited to rotation and translation. For an overlook of these operations, see for example [Joy1999].

The 3x3 matrix representation is the basic rotation representation, but it is generally not used for interpolation or optimisation, as such a matrix has nine parameters to represent three degrees-of-freedom, and is therefore a too costly representation. Furthermore, there is no convenient metric that tells us how "close" two matrix rotations are to one another.

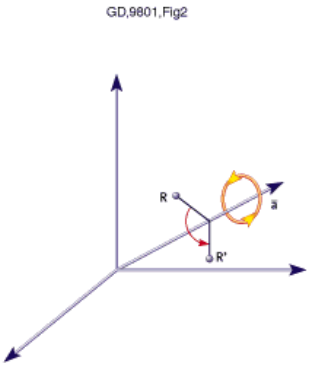


Figure 9.1: Axis-angle rotation from *R* to *R'*, around axis *a* [Bobick1998].

Derivatives The Jacobian of a matrix is easily computed by evaluating the derivative of each element.The Jacobian of a matrix is easily computed by evaluating the partial derivatives of each of the one-axis rotation matrices.

9.1.2 Axis-angles

Axis-angles are perhaps the most intuitive rotation representation, defined as an arbitrary rotation θ around an arbitrary axis v (see Figure 9.1), an axis-angle therefore being represented by four parameters. However, axis-angles require numerical interpolation between two orientations, and do not provide a metric for estimating closeness, on top of being subject to singularities when θ is a multiple of 2Π and having an infinity of representations for zero rotation.

Derivatives The computation of axis-angle derivatives is equivalent to the one for exponential maps, detailed in the following sub-section.

9.1.3 Exponential maps

Exponential maps are derived from axis-angles and based on the observation that only the direction of the axis is important, and not its length, and the representation can therefore be reduced to three parameters as a 3D vector w. The direction of this vector is the axis, and the length is the angle:

$$a = \frac{w}{||w||}, \theta = ||w||$$

The definition of the exponential map follows directly from the corresponding axis-angle:

$$e^w = \left[\frac{\sin(\frac{\theta}{2})}{\theta} w, \cos(\frac{\theta}{2})\right]^T \tag{9.2}$$

Exponential maps of course inherit the singularity of the axis-angle. However, this representation is better suited for optimisation on the one hand, as the number of parameters is reduced to three, and for interpolation on the other, as one can use the geodesic interpolant of quaternions, given that the notation is closely related to this formalism.

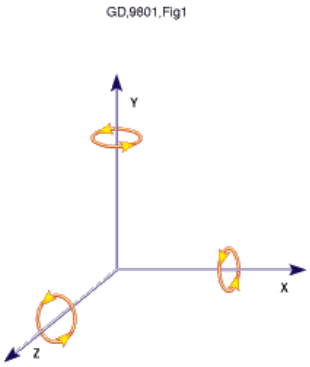


Figure 9.2: Euler angle 1 DOF rotation [Bobick1998].

Derivatives Two methods for computing exponential map derivatives are possible. Either:

1. we use the immediate equivalence with the quaternion notation and use the chain rule, or
2. we directly derive eq.(9.2).

(1) We compute the Jacobians of the equivalent rotation matrix R:

$$\frac{\partial R}{\partial w} = \frac{\partial R}{\partial q}.\frac{\partial q}{\partial w}$$

where $\frac{\partial R}{\partial q}$can be computed from the partial derivatives of R with respect to q_x, q_y, q_z and q_w. As to the second term, $\frac{\partial q}{\partial w}$, its expression is given in [Grassia1998].

(2) The equivalent rotation matrix *R* is obtained by exponiating the matrix $H = [a\times]$ defined as the cross product operator of rotational axis $a = [a_x, a_y, a_z]$:

$$H = [a\times] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{9.3}$$

For the derivation details for this notation, see [Kuehnel2003], the end result being:

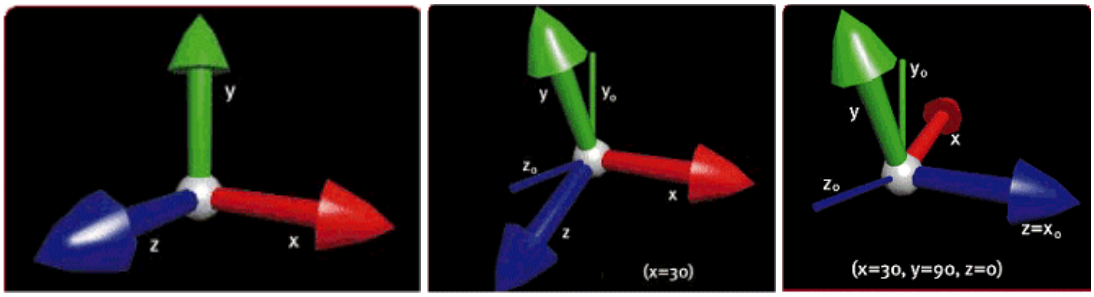
$$\frac{\partial R}{\partial w_\alpha} = \frac{\partial H}{\partial a_\alpha}.\frac{\sin\theta}{\theta} + \frac{\partial H^2}{\partial a_\alpha}.\frac{(1.-\cos\theta)}{\theta} + \left(H(\cos\theta - \frac{\sin\theta}{\theta}) + H^2\left(\sin\theta - 2.\frac{(1.-\cos\theta)}{\theta}\right)\right).a_\alpha$$

where α runs over $\{x, y, z\}$.

9.1.4 Euler angles

The Euler angle representation is the most popular rotation representation. A Euler angle is actually a 1 DOF rotation around one of the co-ordinate system axes. If we want a 3 DOF rotation, we concatenate three such Euler angles, derived directly from the multiplication of three 3x3 rotation matrices (see Figure 9.2).

However, as matrix multiplication is not commutative, the Euler angle representation is not unique, and furthermore, it must be specified whether the axes of the co-ordinate system are static or rotating. If they are static, then performing a rotation first around the x-axis, then around the y-axis, and finally around the z-axis, for example, would mean that the after the first rotation around x is applied, the second rotation around y is



(a) (b) (c)

Figure 9.3: Gimbal lock: (a) Original axis configuration; (b) Configuration after a 30 deg rotation around the x-axis; (c) Configuration after 90 deg rotation around the y-axis: the current z-axis is aligned with the original x-axis, meaning that any further rotation is equivalent to rotating around the x-axis. [Lander1998].

applied to the original orientation of the y-axis, and not to its new orientation following the rotation around x. The bottom-line is that there are 24 possible conventions for Euler angles, thus making it a confusing rotation representation.

If performing the reverse operation, i.e. converting a Euler angle to a rotation matrix, the problem is ill-defined as the solution is not unique [Slabaugh1999].

Furthermore, it is afflicted with the sadly notorious Gimbal lock phenomenon, where following a $\frac{\pi}{2}$or $-\frac{\pi}{2}$around the y axis, one degree of freedom is lost as the x and z axes become aligned [Watt and Watt1992]. This phenomenon is shown in Figure 9.3. This singularity can be explained mathematically by deriving the rotation matrix from the Euler angle $(\theta_x, \theta_y, \theta_z)$ where θ_x and θ_z are arbitrary, and $\theta_y = \frac{\pi}{2}$. We then get:

$$R(\theta_x, \theta_y, \theta_z) = \begin{bmatrix} 0 & 0 & -1 \\ (\sin\theta_x \cos\theta_z - \cos\theta_x \sin\theta_z) & (\sin\theta_x \sin\theta_z + \cos\theta_x \cos\theta_z) & 0 \\ (\cos\theta_x \cos\theta_z + \sin\theta_x \sin\theta_z) & (\cos\theta_x \sin\theta_z - \sin\theta_x \cos\theta_z) & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & -1 \\ \sin(\theta_x - \theta_z) & \cos(\theta_x - \theta_z) & 0 \\ \cos(\theta_x - \theta_z) & \sin(\theta_x - \theta_z) & 0 \end{bmatrix}$$

and as is clearly shown, the resulting transformation now only depends on θ_x and θ_z.

On top of this singularity, Euler angles for 3D rotations do not yield smooth interpolation, as the co-ordinates of each axis are interpolated independently, therefore not taking into account the axes inter-dependencies. Also, they do they provide a very intuitive rotation representation nor a meaningful metric. However, as this representation involves only three parameters, it is widely used for optimisation.

Derivatives The gradient computation for this kind of rotational joint is implemented as in [Craig1989]. For a configuration such as the one shown in Figure 9.4, for observation *x* attached to the end effector *E*, we wish to compute the gradient corresponding to the change induced by a rotation around axis *a* at joint *j*.

This gradient vector is defined as:

$$\frac{\partial}{\partial \theta_j^r} = a \times jx$$

where $[a\times]$ is the cross product operator of the rotational axis defined in eq.(9.3) and *jx* is the vector from joint *j* to observation *x*.

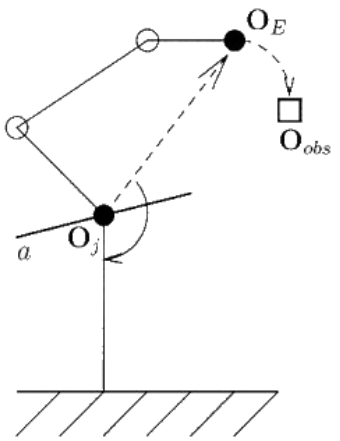


Figure 9.4: Example of an articulated structure, rotating around axis *a* in joint *j*, positioned in O_j. O_E is the end effector to which is attached the observation O_{Obs} [Plänkers2001].

9.1.5 Quaternions

Quaternions have in recent years become more and more popular, since the publication of [Shoemake1985] on the use of quaternions for animating rotations.

A quaternion is defined by its scalar part q_w, and its vectorial part v=[q_x, q_y, q_z]. Any non-zero quaternion can be used for rotation, but in practice, the unit quaternions only are used as these can be related to the notion of axis-angle, given the equivalence:

$$q = \left[\sin\left(\frac{\alpha}{2}\right).\frac{a}{||a||}, \cos\left(\frac{\alpha}{2}\right)\right]$$

where a is the axis of rotation and α the angle of rotation of the axis-angle.

These unit quaternions constitute a hyper-sphere S^3 in quaternion space.

A unit quaternion $q = [s, v]$ is equivalent to a rotation of $\theta \in]-\pi, \pi]$ around v', the normalised v axis, so that $q = [\cos\theta, v' \sin\theta]$ (see Figure 9.5(a)). As we are considering unit quaternions, the scalar component can be deduced from the vectorial part:

$$q_w = \pm\sqrt{1.0 - q_x^2 - q_y^2 - q_z^2} \tag{9.4}$$

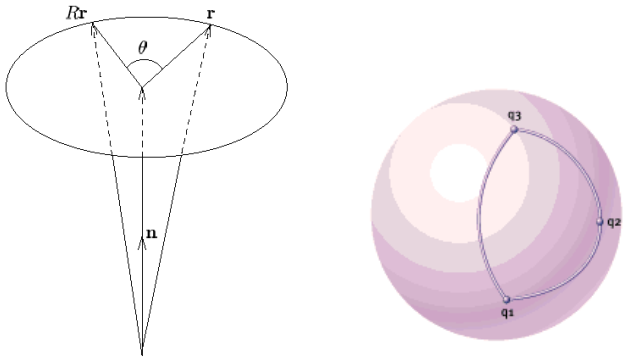
In practical terms, we can view the vectorial part as the axis of rotation, and its length as the amount of rotation, up to a scaling factor.

This representation has therefore four parameters, but on the other hand is not subject to singularities, and allows smooth interpolation on top of providing a straight-forward metric. The interpolation and metric properties derive naturally from the hyper-sphere S^3, as we can perform spherical linear interpolation on the surface of the hyper-sphere (Figure 9.5(b)), and the spherical metric for S^3 is equivalent to the angular metric[1] for SO(3), the space of three-dimensional rotations.

For an excellent and exhaustive report on quaternions and their use in animation, see [Dam *et al.*1998].

In terms of optimisation, as rotations are represented by unit quaternions, the unitary condition needs to be enforced. There are two possible options for carrying out this task:

[1] The angular metric is defined as the cosine between two vectors, hence with values within the interval [-1.,1.].



(a) (b)

Figure 9.5: (a) Quaternion rotation from *r* to *Rr* by angle θ around the axis given by the unit vector *n* [Dam *et al.*1998]; (b) Spherical linear interpolation on the hyper-sphere [Bobick1998].

- optimisation can be done over only three parameters, namely the vectorial part of the quaternion, the disadvantage being that the positivity or negativity of the scalar component q_w is impossible to enforce, as is shown by eq.(9.4);
- otherwise, one can optimise over all four parameters, and then normalise the resulting solution in order to obtain a unit quaternion, though this means computation overhead.

[Schmidt and Niemann2001] outlined a solution to this problem, proposing a new parameterisation for quaternions that makes extensive use of the hyper-sphere structure of unit quaternions S^3. With the new formalism, optimisation is performed on a 3D vector evolving in the tangential hyper-plane to S^3. This parameterisation overcomes the above-mentioned drawbacks, even if it comes at the cost of some additional mathematical manipulations.

Derivatives The computation of quaternion derivatives has been discussed in Section 9.1.3.

9.2 Conversions

9.2.1 Euler angle to 3×3 matrix

We assume that the order of rotation is XYZ, this implying that the single-axis rotation matrices are multiplied in the reverse order. If we wish to convert the Euler angle $e = (e_x, e_y, e_z)$ to its equivalent matrix rotation R, then the single axis-rotation matrices R_x, R_y and R_z are defined as per Section 9.1.1 and we have:

$$R = R_z \star R_y \star R_x$$

This yields the matrix:

$$R = \begin{bmatrix} \cos e_y \cos e_z & \cos e_z \sin e_x \cos e_y - \cos e_x \sin e_z & \cos e_x \cos e_z \sin e_y + \sin e_x \sin e_z \\ \cos e_y \sin e_z & \cos e_x \cos e_z + \sin e_x \sin e_y \sin e_z & -\cos e_z \sin e_x + \cos e_x \sin e_y \sin e_z \\ -\sin e_y & \cos e_y \sin e_x & \cos e_x \cos e_y \end{bmatrix} \quad (9.5)$$

9.2.2 3×3 matrix to Euler angle

To determine the Euler angle $e = (e_x, e_y, e_z)$ from a rotation matrix R, we proceed as follows. From eq.(9.5), we directly obtain the value of R_{31}, the matrix configuration being as in eq.(9.1). By isolating $\cos e_y$, we obtain a set of equations that are solved for e_y. The obtained value of e_y is situated within the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$, under the assumption that $\cos e_y$ is positive. We refer the reader interested in the expression of all the possible matrix to Euler angle decompositions, without *a priori* sign assumptions, to publication [Slabaugh1999].

The equations for the $\cos$ and $\sin$ values of e_x, e_y and e_z are:

$$e_y = \arcsin(-R_{31})$$

$$\cos e_x = \frac{R_{33}}{\cos e_y}$$

$$\sin e_x = \frac{R_{32}}{\cos e_y}$$

$$\cos e_z = \frac{R_{11}}{\cos e_y}$$

$$\sin e_z = \frac{R_{21}}{\cos e_y}$$

under the condition that $\cos e_y \neq 0$. If $\cos e_y = 0$, then we have $e_y = \pm\frac{\pi}{2}$, which corresponds to Gimbal lock, thus making aligning the x-axis with the z-axis, as shown in Section 9.1.4. In this case, we define $e_z = 0$, in which case the angles become:

$$e_y = \arcsin(-R_{31})$$

$$\cos e_x = R_{22}$$

$$\sin e_x = -R_{23}$$

$$e_z = 0$$

9.2.3 Quaternion to 3×3 matrix

Converting a unit quaternion $q = (q_x, q_y, q_z, q_w)$ to its equivalent rotation matrix R yields:

$$R = \begin{bmatrix} 1. - (2.q_y q_y + 2.q_z q_z) & 2.q_x q_y - 2.q_w q_z & 2.q_x q_z + 2.q_w q_y \\ 2.q_x q_y + 2.q_w q_z & 1. - (2.q_x q_x + 2.q_z q_z) & 2.q_y q_z - 2.q_w q_x \\ 2.q_x q_z - 2.q_w q_y & 2.q_y q_z + 2.q_w q_x & 1. - (2.q_x q_x + 2.q_y q_y) \end{bmatrix}$$

9.2.4 3×3 matrix to quaternion

Converting a rotation matrix R to its equivalent unit quaternion $q = (q_x, q_y, q_z, q_w)$ gives us:

$$q_w = \pm\frac{\sqrt{R_{11} + R_{22} + R_{33} + 1.}}{2}$$

$$q_x = \frac{R_{32} - R_{23}}{4.q_w}$$

$$q_y = \frac{R_{13} - R_{31}}{4.q_w}$$

$$q_z = \frac{R_{21} - R_{12}}{4.q_w}$$

The sign of q_w cannot be determined and must be chosen arbitrarily, the signs of q_x, q_y and q_z changing accordingly.

9.2.5 Euler angle to quaternion

If we wish to convert the Euler angle $e = (e_x, e_y, e_z)$ to its equivalent unit quaternion $q = (q_x, q_y, q_z, q_w)$, we carry this out by multiplying single axis quaternion rotations:

$$q = Q_x * Q_y * Qz$$

where

$$Q_x = [\sin\frac{e_x}{2.}, 0., 0., \cos\frac{e_x}{2.}],\ Q_y = [0., \sin\frac{e_y}{2.}, 0., \cos\frac{e_y}{2.}],\ Q_z = [0., 0., \sin\frac{e_z}{2.}, \cos\frac{e_z}{2.}]$$

9.2.6 Quaternion to Euler angle

This conversion is carried out by first converting the quaternion to its equivalent rotation matrix decomposition, and then converting the latter to a Euler angle. Note that the matrix to Euler angle conversion is in general not unique.

9.3 Least-squares optimisation

Least-squares optimisation is a procedure that aims at finding the best-fitting curve to a set of points by minimising the sum of the squares of the offsets, the offset being a defined distance function from the curve to the points. The sum of the squares is used instead of just the offset value so that the function to be minimised is continuously differentiable. The drawback of using the square distance is that outliers can have a disproportionate effect on the fit.

9.3.1 Non-linear least squares fitting

Say we have a function $f(x)$ that depends on n parameters x_n and has a known analytical form. Say that m values of $f(x)$ are known, so that we have an over-determined system of equations such as:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$

$$...$$

$$a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n = b_m$$

where $m \gg n$, meaning we have more unknown variables a_{ij} than observations b_i.

In matrix form, the above equations yields the equality $Ax = b$, A being an $m \times n$ matrix. We would like to find a vector x so that $||Ax - b||$ is minimal.

A vector x is solution to this equation if and only if $A^T Ax = A^T b$, which follows directly from the expression of the square error function F:

$$F(x) = ||Ax - b||^2 = (Ax - b)^T (Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b$$

whose minima are $F'(x) = 0 = 2(A^T Ax - A^T b)$.

Setting the partial derivatives to 0 produces estimating equations for the a_{ij} coefficients. As these equations are in general non-linear, numerical optimisation schemes are therefore required, such as the Gauss-Newton method.

9.3.2 The Levenberg-Marquardt method (damped least-squares)

If for the above system of equations, we can compute not only the Jacobian matrix of first order derivatives, but also the Hessian matrix (i.e. the matrix of partial second derivatives), then more efficient algorithms for locating the minima can be used, such as the Levenberg-Marquardt method that takes advantage of the fact that the Hessian matrix can be generated directly if the function $f(x)$ is twice-differentiable.

If A is a good approximation to the minimum, then we update our current estimate of the minimum x by using $x \to x - A^{-1}b$. If, on the other hand, we are far away from the minimum, then we need to move in the other directly by a factor β, i.e. $x \to x - \beta b$.

To perform this, a new $n \times n$ matrix A' is defined as:

$$A'_{ii} = (1 + \lambda)A_{ii}, \ (i = 1..n)$$

$$A'_{ij} = A_{ij}, \ (i \neq j)$$

where λ is a regularisation constant. At any point x, the suggested next point is given by $(x + x')$ where x' is the solution to $A'x = b$. When λ is small, we have $A' \approx A$ and we move towards the minima. If λ is large, x' gives a guess for a downhill step.

If we start with a small value of λ, say 0.001, the algorithm proceeds as follows:

1. Solve for x' and evaluate $f(x + x')$.
2. If $f(x + x') \geq f(x)$ then increase λ by a factor 10.
3. If $f(x + x') < f(x)$ then decrease λ by a factor 10 and update the current solution $x \to x + x'$.

The algorithm iterates until successive iterations only yield a small improvement in the value of $f(x)$. Iterations can nevertheless remain stuck in a local minima, if modification of the value of λ in either direction does not decrease the square error. If λ were allowed to increase or decrease by a factor larger than 10, it might be possible to remedy this problem.

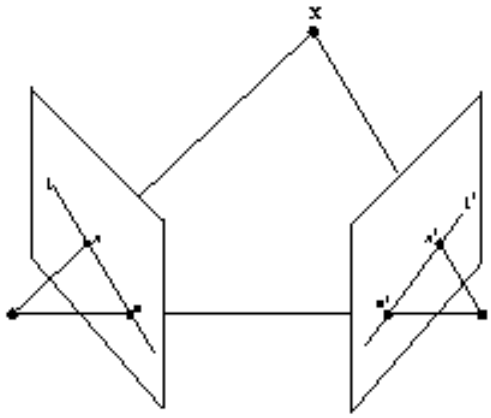


Figure 9.6: Epipolar geometry of two cameras.

9.4 Stereo vision

The principle of stereo vision consists in the reconstruction of 3D data from 2D data using stereo triangulation [Faugeras and Robert1994]. Given the projection P_1 of a point in an image, the corresponding projection in another image must lie on the epipolar line. The correspondence between the two images is established by the fundamental matrix, computed on the basis of the projection matrices themselves derived from the intrinsic parameters of the cameras, obtained during calibration. This matrix establishes the correspondences between an image point in view 1 and the related image point in view 2. In this manner, we calculate the equation of the epipolar line using

$$m'^T.F.m = 0$$

where m' is the vector of the image point in the second camera view, F the fundamental matrix, and m the vector of the image point in the first camera view. As can be seen in Figure 9.6, the matching often results in ambiguities, as there might be several candidate correspondences on the line. Such cases are readily disambiguated if at least a third view exists. If this should not be the case, inevitable depth ambiguities arise.

Bibliography

[Aggarwal and Cai1999] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding: CVIU*, 73(3):428–440, 1999.

[Allen *et al.*2003] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. *Proceedings of Siggraph 2003*, 22(3):587–594, 2003.

[Aubel2002] A. Aubel. *Anatomically-Based Human Body Deformations*. PhD thesis, EPFL, Lausanne, Switzerland, 2002.

[Badler *et al.*1993] N. Badler, C.B. Phillips B.L., and Webber. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, 1993.

[Baerlocher and Boulic1998] P. Baerlocher and R. Boulic. Task priority formulations for the kinematic control of highly redundant articulated structures. In *IEEE International Conference on Intelligent Robots and Systems*, pages 323–329, 1998. Victoria, Canada.

[Baerlocher and Boulic2000] P. Baerlocher and R. Boulic. Parameterization and range of motion of the ball-and-socket joint. In *Proceedings of Avatars 2000*, 2000.

[Baerlocher2001] P. Baerlocher. *Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures*. PhD thesis, EPFL, Lausanne, Switzerland, 2001.

[Bao and Willems1999] H. Bao and P.Y. Willems. On the kinematic modelling and the parameter estimation of the human shoulder. *Journal of Biomechanics*, 32(9):943–950, 1999.

[Barr1981] A. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[BBC2003] BBC. British broadcasting company science. http://www.bbc.co.uk/science/humanbody/, 2003.

[Biedenharn and Louck1981] L.C. Biedenharn and J.D. Louck. Angular momentum in quantum physics: theory and application. In *Encyclopaedia of Mathematics and Its Applications*. Addison-Wesley Reading, 1981.

[Biryukova *et al.*2000] E.V. Biryukova, A. Roby-Brami, A.A. Frolov, and M. Mokhtari. Kinematics of human arm reconstructed from spatial tracking system recordings. *Journal of Biomechanics*, 33(8):985–995, 2000.

[Bittar *et al.*1995] E. Bittar, N. Tsingos, and M.P. Gascuel. Automatic reconstruction of unstructured 3D data: Combining medial axis and implicit surfaces. *Computer Graphics Forum*, 14(3):457–468, 1995.

[Bloomenthal1990] Jules Bloomenthal. Calculation of reference frames along a space curve. In Andrew Glassner, editor, *Graphics Gems*, pages 567–571. Academic Press, Cambridge, MA, 1990.

[Bobick1998] N. Bobick. Rotating objects using quaternions. *Game Developer*, 2, Issue 26, 1998.

[Bodenheimer *et al.*1997] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture: Dealing with the data. In *Eurographics Workshop on Computer Animation and Simulation*, pages 3–18, 1997.

[Bottino *et al.*1998] A. Bottino, A. Laurentini, and P. Zuccone. Toward non-intrusive motion capture. In *Asian Conference on Computer Vision*, pages 416–423, 1998.

[Boulic *et al.*1997] R. Boulic, R. Mas, and D. Thalmann. Interactive identification of the center of mass reachable space for an articulated manipulator. In *International Conference of Advanced Robotics*, pages 589–594, 1997.

[Breen and Whitaker2001] D. Breen and R. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 2001.

[Bregler and Malik1998a] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *CVPR*, 1998.

[Bregler and Malik1998b] C. Bregler and J. Malik. Video motion capture. In *SIGGRAPH*, 1998.

[Cappozzo *et al.*1996] A. Cappozzo, F. Catani, A. Leardini, M.G. Benedetti, and U. Della Croce. Position and orientation in space of bones during movement: experimental artefacts. *Clinical Biomechanics*, 10:90–100, 1996.

[Carr *et al.*2001] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of Siggraph 2001*, volume 2, 2001.

[Carranza *et al.*2003] J. Carranza, C. Theobalt, M.A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *Proceedings of Siggraph 2003*, 22(3):569–576, 2003.

[Cerveri *et al.*2003] P. Cerveri, M. Rabufetti, A. Pedotti, and G. Ferrigno. Real-time human motion estimation through biomechanical models and non-linear state-space filters. *Medical and Biological Engineering and Computing*, 41(2):109–123, 2003.

[Cham and Rehg1998] T.-J. Cham and J. M. Rehg. A multiple hypothesis approach to figure tracking. In *Perceptual User Interfaces,*, pages 19–24, November 1998.

[Cheung *et al.*2003] K.M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[Choo and Fleet2001] K. Choo and D. Fleet. People tracking using hybrid monte carlo filtering. In *ICCV*, pages 321–328, 2001.

[Chu *et al.*2003] C.-W. Chu, O. C. Jenkins, and M. J. Mataric. Markerless kinematic model and motion capture from volume sequences. In *CVPR*, 2003.

[Chua *et al.*2002] C. S. Chua, H. Guan H., and Y.K. Ho. Model based 3d hand posture estimation from a single 2d image. *Image and Vision Computing*, 20:191–202, 2002.

[Consortium1997] The Web3D Consortium. The humanoid animation specification (h-anim). http://H-Anim.org, 1997.

[Cook *et al.*1997] B. Cook, D. Asimov, and D. Hurley. Theory and computational methods for dynamic projections in high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 1997.

[Craig1989] J.J. Craig. *Introduction to Robotics Mechanics and Control*. Addison-Wesley, 1989.

[Dam *et al.*1998] E. Dam, M. Koch, and M. Lillholm. Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, Denmark, 1998.

[D'Apuzzo *et al.*2000] N. D'Apuzzo, A. Gruen, R. Plänkers, and P. Fua. Least squares matching tracking algorithm for human body modeling. In *XIX Congress of the International Society for Photogrammetry and Remote Sensing*, 2000.

[Davison *et al.*2001] A. Davison, J. Deutscher, and I. Reid. Markerless motion capture of complex full-body movement for character animation. In *Eurographics Workshop on Animation and Simulation*, 2001.

[DeGroot and Brand2001] J.H. DeGroot and R. Brand. A three dimensional regression model of the shoulder rhythm. *Journal of Clinical Biomechanics*, 16(9):735–743, 2001.

[DeGroot1997] J.H. DeGroot. The variability of shoulder motions recorded by means of palpation. *Journal of Clinical Biomechanics*, 12(7/8):461–472, 1997.

[Demirdjian2003] D. Demirdjian. Enforcing constraints for human body tracking. In *Workshop on Multi-Object Tracking*, 2003.

[Deutscher *et al.*2000] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 126–133, 2000.

[DiFranco *et al.*2001] D.E. DiFranco, T.-J. Cham, and J.M. Rehg. Reconstruction of 3d figure motion from 2d correspondences. In *CVPR*, 2001.

[Dockstader and Tekalp2002] S. Dockstader and A. M. Tekalp. A kinematic model for human motion and gait analysis. In *ECCV Workshop on Statistical Methods in Video Processing*, 2002.

[Drill1992] M. Drill. Insite fitness. http://www.insitefitness.com.au/, 1992.

[Drummond and Cipolla2001] T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *ICCV*, pages 315–320, 2001.

[Edelsbrunner and Mücke1994] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[Edelsbrunner1999] Herbert Edelsbrunner. Deformable smooth surface design. *Discrete Computational Geometry*, 21:87–115, 1999.

[Engin and Chen1986] A.E. Engin and S.M. Chen. Statistical data base for the biomechanical properties of the human shoulder complex - i: Kinematics of the shoulder complex. *Journal of Biomechanical Engineering*, 108:215–221, 1986.

[Engin and Tümer1989a] A.E. Engin and S.T. Tümer. Three-dimensional kinematic modeling of the human shoulder complex. *Journal of Biomechanical Engineering*, 111:107–121, 1989.

[Engin and Tümer1989b] A.E. Engin and S.T. Tümer. Three-dimensional kinematic modeling of the human shoulder complex-part i: Physical model and determination of joint sinus cones. *Journal of Biomechanical Engineering*, 111:107–112, 1989.

[Engin and Tümer1989c] A.E. Engin and S.T. Tümer. Three-dimensional kinematic modeling of the human shoulder complex-part ii: Mathematical modelling and solution via optimization. *Journal of Biomechanical Engineering*, 111:113–121, 1989.

[Faugeras and Robert1994] Olivier D. Faugeras and Luc Robert. What can two images tell us about a third one? In *ECCV (1)*, pages 485–492, 1994.

[Feiner and Beshers1990] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of UIST '90, Snowbird, Utah*, pages 76–83, October 1990.

[Ferrigno and Pedotti1985] G. Ferrigno and A. Pedotti. Elite: a digital dedicated hardware system for movement analysis via real-time tv signal processing. *Transactions on Biomedical Engineering*, 32:943–950, 1985.

[Gavrila and Davis1995] D.M. Gavrila and L.S. Davis. 3-D Model-based Tracking of Human Upper Body Movement: a Multi-View Approach. In *IEEE Int. Symposium on Computer Vision*, pages 253–258. IEEE Computer Society Press, November 1995.

[Gavrila1999] D.M. Gavrila. The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, 73(1), January 1999.

[Gomes and Mojsilovic2002] Jose Gomes and Aleksandra Mojsilovic. A variational approach to recovering a manifold from sample points. In *ECCV (2)*, pages 3–17, 2002.

[Goncalves *et al.*1995] L. Goncalves, E.D. Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3d. In *ICCV*, pages 764–770, 1995.

[Grassia1998] F.S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[Hamill and Knutzen1995] J. Hamill and K.M. Knutzen. *Biomechanical Basis of Human Movement*. Williams & Wilkins, 1995.

[Hanson and Ma1995] A.J. Hanson and H. Ma. Quaternion frame approach to streamline visualization. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):164–174, June 1995.

[Hanson1998a] A.J. Hanson. Constrained optimal framings of curves and surfaces using quaternion gauss maps. In *Proceedings of Visualization '98*, pages 375–382. IEEE Computer Society Press, 1998.

[Hanson1998b] A.J. Hanson. Quaternion gauss maps and optimal framings of curves and surfaces. Technical Report 518, Indiana University Computer Science Department, October 1998.

[Heap and Hogg1996] T. Heap and D. Hogg. Towards 3d hand tracking using a deformable model. In *Conference on Automatic Face and Gesture Recognition*, pages 140–145, 1996.

[Högfors *et al.*1991] C. Högfors, B. Peterson, G. Sigholm, and P. Herberts. Biomechanical model of the human shoulder - ii: The shoulder rhythm. *Journal of Biomechanics*, 24(8):699–709, 1991.

[Holt *et al.*1997] R. J. Holt, T. S. Huang, A. N. Netravali, and R. J. Qian. Determining articulated motion from perspective views: A decomposition approach. *Pattern Recognition*, 30:1435–1449, 1997.

[Hunter *et al.*1997] E. Hunter, P. Kelly, and R. Jain. Estimation of articulated motion using kinematically constrained mixture densities. In *IEEE Nonrigid and Articulated Motion Workshop*, 1997.

[Ilic and Fua2003] S. Ilic and P. Fua. From explicit to implicit surfaces for visualization, animation and modeling. In *ISPRS Workshop on Visualization and Animation of Reality-based 3D objects*, 2003.

[Isard and Blake1998] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[IST1999] European Commission / IST. Virtual animation of the kinematics of the human for industrial, educational and research purposes. http://www.ulb.ac.be/project/vakhum/, 1999.

[Iwasawa1997] S. Iwasawa. Real-time estimation of human body posture from monocular thermal images. In *CVPR*, pages 15–20, 1997.

[Jin *et al.*2000] X. Jin, Y.F. Li, and Q. Peng. General constrained deformation based on generalized metaballs. *Computers and Graphics*, 24(2):219–231, 2000.

[Joy1999] Kenneth I. Joy. How transformations work. http://graphics.cs.ucdavis.edu/GraphicsNotes/How-Transformations-Work.pdf, 1999.

[Kakadiaris and Metaxas1995] I. Kakadiaris and D. Metaxas. 3d human body model acquisition from multiple views. In *IEEE International Conference on Computer Vision*, pages 618–623, 1995.

[Kakadiaris and Metaxas1996] I. A. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 81–87, Los Alamitos, California, U.S.A., 18–20 1996. IEEE Computer Society.

[Kakadiaris and Metaxas2000] I. A. Kakadiaris and D.N. Metaxas. Model-based estimation of 3d human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459, 2000.

[Kakadiaris *et al.*1994] I. A. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. In *Proc. IEEE Computer Vision and Pattern Recognition Conference*, Seattle, Washington, U.S.A., 1994. IEEE Computer Society.

[Kameda *et al.*1993] Y. Kameda, M. Minoh, and K. Ikeda. Three dimensional pose estimation of an articulated object from its silhouette image. In *Proceedings of Asian Conference on Computer Vision*, pages 612–615, 1993.

[Kass *et al.*1987] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. of IEEE Conference on Computer Vision*, pages 259–268, London, England, 8-11 1987.

[Klein Breteler *et al.*1999] M.D. Klein Breteler, C.W. Spoor, and F.C.T. VanDerHelm. Measuring muscle and joint geometry parameters of a shoulder for modeling purposes. *Journal of Biomechanics*, 32(11):1191–1197, 1999.

[Korein1985] J.U. Korein. *A Geometric Investigation Of Reach.* MIT Press, Cambridge, 1985.

[Kuch and Huang1995] J. Kuch and T. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *ICCV*, pages 666–671, 1995.

[Kuehnel2003] Frank O. Kuehnel. On the minimization over so(3) manifolds. Technical Report 00000109, Research Institute for Advanced Computer Science, California, USA, May 2003.

[Lander1998] J. Lander. Better 3d: The writing is on the wall. *Game Developer*, pages 15–21, March 1998.

[Lawton and Beard2001] J. Lawton and R. Beard. Model independent approximate eigenaxis rotations via quaternion feedback. Technical report, Brigham Young University, Utah, USA, 2001.

[Lazarus and Verroust1998] Francis Lazarus and Anne Verroust. 3d metamorphosis: a survey. *The Visual Computer*, 14(8/9):373–389, 1998.

[Lee and Kunii1993] J. Lee and T. Kunii. Constraint-based hand animation. In *Models and techniques in computer animation*, pages 110–127. Springer Verlag, Tokyo, Japan, 1993.

[Lee2000] J. Lee. *A Hierarchical Approach to Motion Analysis and Synthesis for Articulated Figures*. PhD thesis, Department of Computer Science, KAIST, 2000.

[Leonardis *et al.*1997] A. Leonardis, A. Jaklic, and F. Solina. Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1289–1295, 1997.

[Lin *et al.*2001] J. Lin, Y. Wu, and T.S. Huang. Modeling the constraints of human hand motion. In *5th Annual Federated Laboratory Symposium*, 2001.

[Liu *et al.*1999] X. Liu, Y. Zhuang, Y. Wu, and Y. Pan. Video based human motion capture. In *IEEE Signal Processing Society 1999 Workshop on Multimedia Signal Processing*, 1999.

[Lorensen and Cline1987] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):163–169, 1987.

[Luttgens and Hamilton1997] K. Luttgens and N. Hamilton. Kinesiology. In *Scientific Basis of Human Motion*. Brown and Benchmark, Madison, Wisconsin, USA, 1997.

[Malik *et al.*1993] N. Malik, T. Dracos, and D. Papantoniou. Particle tracking in three-dimensional turbulent flows - part ii: Particle tracking. *Experiments in Fluids*, 15:279–294, 1993.

[Maurel and Thalmann2000] W. Maurel and D. Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers and Graphics*, 24(2), 2000.

[Maurel1998] W. Maurel. *3D Modeling of the Human Upper Limb including the Biomechanics of Joints, Muscles and Soft Tissues*. PhD thesis, EPFL, Lausanne, Switzerland, 1998.

[Menache1999] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann, 1999.

[Meskers *et al.*1998] C.G.M. Meskers, H.M. Vermeulen, J.H. DeGroot, F.C.T. VanDerHelm, and P.M. Rozing. 3d shoulder position measurements using a six-degree-of-freedom electromagnetic tracking device. *Clinical Biomechanics*, 13:280–292, 1998.

[Meskers *et al.*1999] C.G.M. Meskers, H. Fraterman, F.C.T. VanDerHelm, H.M. Vermeulen, and P.M. Rozing. Calibration of the flock of birds electromagnetic tracking device and its application in shoulder motion studies. *Journal of Biomechanics*, 32(6):629–633, 1999.

[Metaxas and Terzopoulos1993] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.

[Mikic *et al.*2003] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–267, 2003.

[Moeslund and Granum2000] T. Moeslund and E. Granum. Multiple cues used in model-based human motion capture. In *4th International Conference on Automatic Face and Gesture Recognition, Grenoble, France*, 2000.

[Moeslund and Granum2001] T.B. Moeslund and E. Granum. A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding*, 81, March 2001.

[Moeslund *et al.*2002] T.B. Moeslund, M. Vittrup, K. Skov Pedersen, M. Klinge Laursen, M.K. Damgaard Sørensen, H. Uhrenfeldt, and E. Granum. Estimating the 3d shoulder position using monocular vision and a detailed shoulder model. In *International Conference on Imaging Science, Systems, and Technology, Las Vegas, Nevada*, 2002.

[Moeslund *et al.*2003] T.B. Moeslund, C.B. Madsen, and E. Granum. Modelling the 3d pose of a human arm and the shoulder complex utilising only two parameters. In *International Conference on Model-based Imaging, Rendering, image Analysis and Graphical special Effects, INRIA Rocquencourt, France*, 2003.

[Moeslund2003] T.B. Moeslund. *Computer Vision-Based Motion Capture of Body Language*. PhD thesis, Aalborg University, Aalborg, Denmark, June 2003.

[Molet *et al.*1996] T. Molet, R. Boulic, and D.Thalmann. A real-time anatomical converter for human motion capture. In *7th Eurographics Workshop on Animation and Simulation*, 1996.

[Morris and Rehg1998] D. Morris and J. Rehg. Singularity Analysis for Articulated Object Tracking. In *CVPR*, pages 289–296, 1998.

[Morse *et al.*2001] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 89–98, 2001.

[Munkelt *et al.*1998] O. Munkelt, C. Ridder, D. Hansel, and W. Hafner. A model driven 3d image interpretation system applied to person detection in video images. In *International Conference on Pattern Recognition*, 1998.

[Muraki1991] S. Muraki. Volumetric shape description of range data using "Blobby Model". *Computer Graphics*, 25(4):227–235, 1991.

[Noma *et al.*1999] T. Noma, K. Oishi, H. Futsuhara, H. Baba, T. Ohashi, and T. Ejima. Motion generator approach to translating human motion from video to animation. In *The Seventh Pacific Conference on Computer Graphics and Applications*, 1999.

[O'Brien *et al.*2000] J.F. O'Brien, B. Bodenheimer, G. Brostow, and J. Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphics Interface*, pages 53–60, 2000.

[Ong and Gong1999] E.-J. Ong and S. Gong. Tracking hybrid 2d-3d human models from multiple views. In *International Workshop on Modelling People*, 1999.

[Opalach and Maddock1995] A. Opalach and S. C. Maddock. An overview of implicit surfaces. In *Introduction to Modelling and Animation Using Implicit Surfaces*, pages 1.1–1.13, 1995. Leeds, UK. Course Notes No 3.

[O'Rourke and Badler1980] J. O'Rourke and N.I. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):522–536, 1980.

[Otani1989] E. Otani. Software tools for dynamic and kinematic modeling of human motion. Technical Report MS-CIS-89-43, Computer and Information Science, University of Pennsylvania, Philadelphia, USA, 1989.

[Perales and Torres1994] F.J. Perales and J. Torres. A system for human motion matching between synthetic and real images based on a biomechanic graphical model. In *Workshop on Motion of Non-Rigid and Articulated Objects*, Austin, Texas, U.S.A., 1994.

[Plänkers and Fua2001] R. Plänkers and P. Fua. Articulated soft objects for video-based body modeling. In *IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[Plänkers2001] R. Plänkers. *Human Body Modeling from Video Sequences*. PhD thesis, EPFL, Lausanne, Switzerland, November 2001.

[Ranjan and Fournier1995] V. Ranjan and A. Fournier. Shape transformations using union of spheres. Technical Report TR-95-30, Department of Computer Science, University of British Columbia, 1995.

[Rehg and Kanade1994] James M. Rehg and Takeo Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *ECCV (2)*, pages 35–46, 1994.

[Rehg and Kanade1995] James M. Rehg and Takeo Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, pages 612–617, 1995.

[Rehg *et al.*2003] J. M. Rehg, D. D. Morris, and T. Kanade. Ambiguities in Visual Tracking of Articulated Objects using 2–D and 3–D Models. *International Journal of Robotic Research*, 22(6):393–418, 2003.

[Ren and Xu2001] H. Ren and G. Xu. Articulated-model based upper-limb pose estimation. In *International Symposium on Computational Intelligence in Robotics and Automation*, pages 450–454, 2001.

[Ringer and Lasenby2000] M. Ringer and J. Lasenby. Modelling and tracking articulated motion from multiple camera views. In *11th British Machine Vision Conference*, 2000.

[Ringer and Lasenby2002a] M. Ringer and J. Lasenby. Multiple hypothesis tracking for automatic visual motion capture. In *ECCV*, 2002.

[Ringer and Lasenby2002b] M. Ringer and J. Lasenby. A procedure for automatically estimating model parameters in optical motion capture. In *13th British Machine Vision Conference*, 2002.

[Rosales and Sclaroff2000] Romer Rosales and Stan Sclaroff. Inferring body pose without tracking body parts. In *CVPR*, 2000.

[Saff and Kuijlaars1997] E.B. Saff and A. Kuijlaars. Distributing many points on the sphere. *The Mathematical Intelligencer*, 19:5–11, 1997.

[Schmidt and Niemann2001] J. Schmidt and H. Niemann. Using Quaternions for Parametrizing 3–D Rotations in Unconstrained Nonlinear Optimization. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, pages 399–406, Stuttgart, Germany, 2001. AKA/IOS Press, Berlin, Amsterdam.

[Segawa *et al.*2000] H. Segawa, H. Shioya, N. Hiraki, and T. Totsuka. Constraint-conscious smoothing framework for the recovery of 3d articulated motion from image sequences. In *Automatic Face and Gesture Recognition*, 2000.

[Shoeb1998] M. Shoeb. Improved marching cubes, 1998.

[Shoemake1985] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics*, volume 19, pages 245–254, 1985. Proceedings of SIGGRAPH 1985.

[Silaghi *et al.*1998] Marius-Calin Silaghi, Ralf Plänkers, Ronan Boulic, Pascal Fua, and Daniel Thalmann. Local and global skeleton fitting techniques for optical motion capture. In *Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 26–40, 1998.

[Slabaugh1999] G. Slabaugh. Computing euler angles from a rotation matrix. http://users.ece.gatech.edu/ slabaugh/personal/research/euler/euler.pdf, August 1999.

[Sminchisescu and Telea2002] Cristian Sminchisescu and Alexandru Telea. Human pose estimation from silhouettes. a consistent approach using distance level sets. In *WSCG International Conference on Computer Graphics,Visualization and Computer Vision*, 2002.

[Sminchisescu and Triggs2003] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 2003.

[Tenenbaum *et al.*2000] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[Theobalt *et al.*2002] C. Theobalt, M. Magnor, P. Schueler, and H.P. Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. In *Pacific Graphics*, pages 96–103, 2002.

[Tolani *et al.*2000] D. Tolani, N. Badler, and J. Tallier. A kinematic model of the human arm using triangular bézier spline surfaces. http://citeseer.nj.nec.com/315176.html, 2000.

[Treece *et al.*2001] Graham Treece, Richard Prager, and Andrew Gee. Volume-based three-dimensional metamorphosis using sphere-guided region correspondence. *The Visual Computer*, 17(7):397–414, 2001.

[Tsingos *et al.*1995] N. Tsingos, E. Bittar, and M.P. Gascuel. Implicit surfaces for semi-automatic medical organs reconstruction. In *Computer Graphics International'95*, pages 3–15, Leeds, UK, 1995.

[Turk and O'Brien1999] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. *Computer Graphics*, 33(Annual Conference Series - SIGGRAPH 99):335–342, 1999.

[Turk and O'Brien2000] H. Turk and J.F. O'Brien. Modelling with implicit surfaces that interpolate, 2000.

[Ude and Riley1999] A. Ude and M. Riley. Prediction of body configurations and appearance for model-based estimation of articulated human motions. In *International conference on systems, man, and cybernetics*, pages 62–71, 1999.

[University of Iowa1992] USA University of Iowa. Virtual hospital. www.vh.org, 1992.

[VanDerHelm and Pronk1995] F.C.T. VanDerHelm and G.M. Pronk. Three-dimensional recording and description of motions of the shoulder mechanism. *Journal of Biomechanical Engineering*, 117(1):27–40, 1995.

[VanDerHelm1997] F.C.T. VanDerHelm. A standardized protocol for motion recordings of the shoulder. In *1st Conference of the International Shoulder Group, Maastricht, Netherlands*, pages 7–12, 1997.

[Veldpaus *et al.*1988] F.E. Veldpaus, H.J. Woltring, and L.J.G.M. Dortmans. A least-squares algorithm for the equiform transformation from spatial marker co-ordinates. *Journal of Biomechanics*, 1(21):45–54, 1988.

[Wachter and Nagel1999] S. Wachter and H. Nagel. Tracking persons in monocular image sequences. *Computer Vision and Image Understanding*, 74(3):174–192, 1999.

[Wang *et al.*1998] X. Wang, M. Maurin, F. Mazet, N. De Castro Maia, K. Voinot, J.P. Verriest, and M. Fayet. Three-dimensional modelling of the motion range of axial rotation of the upper arm. *Journal of Biomechanics*, 31(10):899–908, 1998.

[Watson1981] D.F. Watson. Computing the n-dimensional delaunay tesselation with application to voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.

[Watson1992] D.F. Watson. *Contouring — A Guide to the Analysis and Display of Spatial Data*. Pergamon Press, 1992.

[Watt and Watt1992] A. Watt and M. Watt. Advanced animation and rendering techniques, 1992.

[Wilhelms *et al.*2000] J. Wilhelms, A. Van Gelder, L. Atkinson-Derman, and A. Luo. Human motion from active contours. In *Workshop on Human Motion, Austin, Texas, U.S.A.*, 2000.

[Woo1990] A. Woo. Fast ray-box intersection. In David Kirk, editor, *Graphics Gems*, pages 395–396. Academic Press, Cambridge, MA, 1990.

[Wu and Huang1999] Y. Wu and T.S. Huang. Capturing articulated human hand motion: A divide-and-conquer approach. In *ICCV*, pages 606–611, 1999.

[Wyvill and Wyvill1989] B. Wyvill and G. Wyvill. Field functions for implicit surfaces. *Visual Computer*, 5:75–82, 1989.

[Yamamoto and Koshikawa1991] M. Yamamoto and K. Koshikawa. Human motion analysis based on a robot arm model. In *CVPR*, pages 664–665, 1991.

[Yonemoto *et al.*2000] S. Yonemoto, D. Arita, and R. Taniguchi. Real-time human motion analysis and ik-based human figure control. In *Workshop on Human Motion, Austin, Texas, U.S.A.*, 2000.

[Zhao *et al.*2000] H.K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. In *Computer Vision and Image Understanding*, volume 80, pages 295–319, 2000.

[Zhao *et al.*2002] T. Zhao, T. Wang, and H. Shum. Learning a highly structured motion model for 3d human tracking. In *5th Asian Conference on Computer Vision*, 2002.

[Zheng and Suezaki1998] J.Y. Zheng and S. Suezaki. A model based approach in extracting and generating human motion. In *14th International Conference on Pattern Recognition*, volume 2, pages 1201–1205, 1998.

[Zhuang *et al.*1999] Y. Zhuang, X. Liu, and Y. Pan. Video motion capture using feature tracking and skeleton reconstruction. In *Proceeding of IEEE International Conference on Image Processing*, 1999.

Publications

1. R. Boulic and P. Fua and L. Herda and M. Silaghi and J. Monzani and L. Nedel and D. Thalmann, *An Anatomic Human Body for Motion Capture*, in European Multimedia, Microprocessor Systems and Electronic Commerce, Bordeaux, France, 1998.

2. P. Fua and L. Herda and R. Plänkers and R. Boulic, *Full Body Modeling Using Animation Models*, in XXXIII Congress of the International Society for Photogrammetry and Remote Sensing, Amsterdam, The Netherlands, 2000.

3. L. Herda and P. Fua and R. Plänkers and R. Boulic and D. Thalmann, *Skeleton-Based Motion Capture for Robust Reconstruction of Human Motion*, in Proceedings of Computer Animation, Philadelphia, USA, 2000.

4. L. Herda and P. Fua and R. Plänkers and D. and R. Boulic and D. Thalmann, *Using Skeleton-Based Tracking to Increase the Reliability of Optical Motion Capture*, in Human Movement Science, vol. 20, p. 313-341, 2001.

5. L. Herda and R. Urtasun and P. Fua and A.J. Hanson, *An Automatic Method for Determining Quaternion Field Boundaries for Ball-and-Socket Joint Limits*, in Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition, p. 95-100, Washington, USA, 2002.

6. L. Herda and R. Urtasun and P. Fua and A.J. Hanson, *Automatic Determination of Shoulder Joint Limits using Quaternion Field Boundaries*, in International Journal of Robotics Research, vol. 22, no. 6, p. 419-436, 2003.

7. L. Herda and P. Fua, *Hierarchical Implicit Surface Joint Limits to Constrain Video-Based Motion Capture*, submitted to the European Conference on Computer Vision, 2004.

8. L. Herda, R. Urtasun and P. Fua, *Hierarchical Implicit Surface Joint Limits for Human Body Tracking*, submitted to the Human Movement Science journal.

Curriculum vitae

	Lorna HERDA
Born:	18.08.1972 in Sondershausen, Germany
Nationality:	German
Mother tongue:	English and German
Other languages:	Fluent in French, very good knowledge of Spanish, fairly good knowledge of Italian

EDUCATION

1990-1996	MSc in Computer Science, Master's thesis in Operations Research "Creation of a regulation module in the context of airline crew rostering perturbations", University of Geneva, Switzerland
1989-1990	Baccalauréat D (scientific)

PROFESSIONAL EXPERIENCE

2000-2002	Software engineer at Merging Technologies Inc., Puidoux, Switzerland
1996-2000	Information Technology Consultant for the Inter-Governmental Consultations for Asylum, Refugee and Migration Policies, Geneva, Switzerland
1995-1996	Computer Technician for the Inter-Regional Programme for Disabled People, United Nations Development Programme, Geneva, Switzerland